

(19) 世界知的所有権機関
国際事務局



(43) 国際公開日
2005年1月6日 (06.01.2005)

PCT

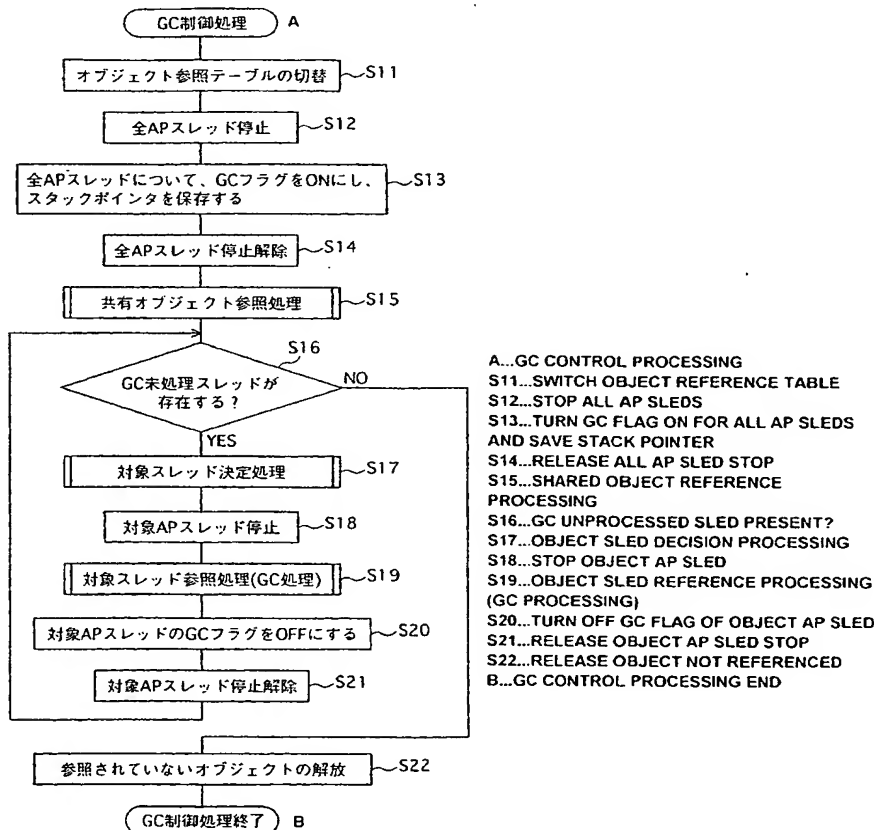
(10) 国際公開番号
WO 2005/001695 A1

- (51) 国際特許分類⁷: G06F 12/00, 9/44 (72) 発明者; および
(21) 国際出願番号: PCT/JP2004/009043 (75) 発明者/出願人(米国についてのみ): 今西 祐子 (IMANISHI, Yuko). 土井 繁則 (DOI, Shigenori).
(22) 国際出願日: 2004年6月21日 (21.06.2004) (74) 代理人: 中島 司朗 (NAKAJIMA, Shiro); 〒5310072 大阪府大阪市北区豊崎三丁目2番1号淀川5番館6F Osaka (JP).
(25) 国際出願の言語: 日本語
(26) 国際公開の言語: 日本語 (81) 指定国(表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.
(30) 優先権データ:
特願2003-187690 2003年6月30日 (30.06.2003) JP
(71) 出願人(米国を除く全ての指定国について): 松下電器産業株式会社 (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.) [JP/JP]; 〒5718501 大阪府門真市大字門真1006番地 Osaka (JP).

[続葉有]

(54) Title: GARBAGE COLLECTION SYSTEM

(54) 発明の名称: ガーベジコレクションシステム



(57) Abstract: It is possible to suppress increase of the time required for garbage collection without prolonging the all-sled stop period of AP. A garbage collection system includes: selection means for successively selecting a plurality of sleds in the execution process of the object-orientated program consisting of a plurality of sleds; inspection means for performing inspection processing for stopping the execution of the sled selected, detecting an object accessible from the sled so as to be managed as a non-release object, and resuming execution of the sled; detection means for detecting that an object pointer is made a processing object by the sled in execution after the start of the selection by the selection means and making the object specified by the object pointer a non-release object; and release means for releasing a memory area corresponding to an object other than the object managed as a non-release object after completion of the inspection processing for all the sleds.

[続葉有]

ATTACHMENT A



(84) 指定国(表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

2文字コード及び他の略語については、定期発行される各PCTガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

添付公開書類:

— 国際調査報告書

(57) 要約:

A P の全スレッド停止期間を長期化させず、ガーベジコレクションに要する時間の増加を抑制することを目的とし、複数スレッドで構成されるオブジェクト指向プログラムの実行過程において、複数スレッドを順に選択する選択手段と、選択されたスレッドの実行を停止し、当該スレッドからアクセス可能なオブジェクトを検出して非解放対象として管理し、当該スレッドの実行を再開する検査処理を実施する検査手段と、前記選択手段による前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象とする検知手段と、前記複数スレッド全てについて前記検査処理が完了した後に非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を開放する解放手段とを備える。

明 細 書

ガーベジコレクションシステム

5 技術分野

本発明は、アプリケーションプログラム（A P : Application Program）により使用されたメモリについてのガーベジコレクション（G C : Garbage Collection）に関する。

10 背景技術

従来のオブジェクト指向のプログラミング言語には、A P の作成者にメモリ領域の確保や解放を意識させず、A P が使用した後に不要となったオブジェクト（「オブジェクトインスタンス」ともいう。）に対応するメモリ領域の解放を実行環境側に任せる方法を採用するものがある。例えばJ a v a 言語である。なお、J

15 a v a は米国S u n M i c r o s y s t e m s 社の商標である。

このような言語を用いて記述されたA P は、不要となったオブジェクトに対応するメモリ領域を自動的に解放するガーベジコレクション（G C）機構を備える実行環境上で動かされる。

G C 機構は、A P の実行に際して動的に確保されたオブジェクトに対応するメモリ領域がどこからも参照されていない状態になっていることを検出してそのメモリ領域を解放し再利用可能状態にする。

マルチスレッド構成のA P において、各スレッドは、それぞれスタック領域と対応し、動作過程において、スタック領域内にデータを格納したり、格納したデータを参照したり、オブジェクトを生成したりする。オブジェクトを生成した場合
25 合には通常はスタック領域内にそのオブジェクトのポインタ、つまりそのオブジェクトのメモリ内における所在位置を指し示すデータ（以下、「オブジェクトポインタ」という。）が格納され、スレッドからのオブジェクトへのアクセスはそのオブジェクトポインタを参照することによりなされる。また、通常、オブジェクトの領域内にも、他のオブジェクトへのオブジェクトポインタが格納される。

30 ある時点においてA P により参照されている全てのオブジェクトは、A P の各

スレッドに対応する各スタック領域内に含まれるオブジェクトポインタから、直接、又は1以上のオブジェクト内のオブジェクトポインタを媒介にして、辿ることができる。

これに対応してGC機構は、基本的に、ある時点においてスタック領域内のオブジェクトポインタから辿れなくなっているオブジェクトに対応するメモリ領域を不要なものとして解放対象にする。

従来のGC方式として知られているマークアンドスイープ方式は、特定のオブジェクトポインタから辿れる全てのオブジェクトにマークを付けていく処理を行った後に、全てのオブジェクトを走査して、マークの付いていないオブジェクトに対応するメモリ領域を解放する方式である。

このマークアンドスイープ方式のGCをマルチスレッド構成のAPの実行時に行う場合に、GCの迅速性を最優先して単純に、全スレッドを停止してから、各スレッドに対応するスタック領域内のオブジェクトポインタから辿れる全てのオブジェクトにマークを付与する処理を行い、その後に全スレッドの停止を解除して、マークの付いていないオブジェクトの解放を行うとするならば、次の問題が生じる。

即ち、APの全スレッドが停止している期間が長期化する可能性があり、この場合に、ユーザの操作等に対して一切反応を示せず、例えばコンピュータのディスプレイの表示内容は変化しないまとなり、ユーザを戸惑わせてしまうという問題である。

この問題を解決する方式として、日本国の特許第3027845号公報には、このマークアンドスイープを用いたGCを、マルチスレッド構成のAPを全く停止することなく実行する方式が提案されている。

この方式は、ルートノードなるオブジェクトから辿れる全てのオブジェクト及びスレッド毎の各スタック領域内のオブジェクトポインタから辿れる全てのオブジェクトに対してマークを付与する第1処理を行い、その第1処理中に、APのスレッド（以下、「APスレッド」という。）の動作によりオブジェクトへのオブジェクトポインタが移動した場合にそのオブジェクトを表すデータをマークスタックなる領域に積んでおき、そのマークを付与する処理が完了した段階で、更にマークスタックから辿れる全てのオブジェクトに対してマークを付与する第2

処理を行い、最後に、マークされていないオブジェクトに対応するメモリ領域を解放するものである。

- しかしながら、上述のAPを全く停止せずにマークを付与する方式では、APスレッドの動作によってスタック領域内のデータが変化するため、第1処理中、
- 5 スタック領域内のオブジェクトポインタから辿れるオブジェクトに対してマークを付与する処理の一部が無駄となる可能性がある。

- 例えば、GCを行うスレッドが、APの1つのスタック領域内のオブジェクトポインタ（ここでは、「オブジェクトポインタA」と名付ける。）を検出して、そのオブジェクトポインタAから辿れるオブジェクトに対してマークを付与する
- 10 処理（ここでは、「処理A」と名付ける。）を実行している間に、そのスタック領域に対応するAPスレッドが、オブジェクトポインタA或いはそれらのオブジェクト内の1又は複数のオブジェクトポインタを、コピーしてスタック領域に新たに格納する動作をしたならば、GCを行うスレッドはその処理Aの終了後にい
- 15 ずれ、再び処理Aと一部重複する処理を行う、或いはその重複防止のためのチェック処理をわざわざ行うことになり、これが無駄となる。この処理の無駄は、GCの開始から完了までに要するCPU時間を無駄に増加させることにつながり、CPUの利用効率を低下させる。

発明の開示

- 20 そこで、本発明は、上記問題に鑑みてなされたものであり、APの全スレッドが停止している期間を長期化させないようにするとともに、GCの開始から完了までに要するCPU時間の無駄な増加をある程度抑制するようなGC方式を用いるガベージコレクション（GC）システムを提供することを目的とする。

- 上記目的を達成するために、本発明に係るGCシステムは、複数スレッドで構成されるオブジェクト指向プログラムの実行過程において不要になったオブジェクトに対応するメモリ領域を解放するGCシステムであって、複数スレッドそれぞれを順に選択する選択手段と、選択されたスレッドについて、当該スレッドの実行を停止して、当該スレッドからオブジェクトポインタの参照を通じてアクセス可能であるオブジェクトを検出して当該検出したオブジェクトを非解放対象として管理して、当該スレッドの実行を再開するという手順からなる検査処理を実
- 30

施する検査手段と、前記選択手段による前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象として管理する検知手段と、前記複数スレッド全てについて前記検査処理が完了した後に
5 いて、非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を、解放する解放手段とを備えることを特徴とする。

ここで、オブジェクトポインタがスレッドにより処理対象にされるということは、CPUによるスレッドの処理過程においてオブジェクトポインタを処理対象とする命令が実行されるということを意味する。

- 10 また、オブジェクトを非解放対象として管理するとは、例えばそのオブジェクトポインタをfromテーブルからtoテーブルに移行する後述の方法等により、マーク付与を実現することをいう。

- これにより、APスレッドからスタック内やオブジェクト内のオブジェクトポインタを介して辿ることができるオブジェクトについての非解放対象としての指
15 定の過程つまりマーク付与の過程においては、そのAPスレッドを停止させているため、そのAPスレッドが動作することに起因してスタック領域内のデータが変化するという事態が起こらないため、その過程におけるマーク付与が無駄になる危険性はなくなり、この点についてはCPUの利用効率低下を防止できる。

- また、これにより、全てのAPスレッドを停止して上述のマーク付与を行って
20 いるのではないため、APの全スレッドが停止している期間の長期化が防止できる。なお、全APスレッドを停止させないことによるオブジェクトへの参照状態の変化に対しては、実行中のスレッドによりオブジェクトポインタが処理対象とされたことを監視して対応しているため、いずれかのスレッドからアクセス可能なオブジェクトについては必ず非解放対象として管理されることになる。

- 25 また、前記検知手段は、前記検知を、実行中のスレッドについて未だ検査処理がなされていない場合に限り行い、前記検知手段は、前記検知をした場合に、当該実行中のスレッドにより処理対象にされたオブジェクトポインタ、及び当該オブジェクトポインタから辿ることのできるオブジェクト内のオブジェクトポインタを、当該スレッドに対応する作業用メモリ領域に保存する検出部と、前記検査
30 手段によりスレッドの実行が停止されている間に、当該スレッドに対応する作業

用メモリ領域内のオブジェクトポインタから辿ることのできるオブジェクトを非解放対象として管理する管理部とを有することとしてもよい。

これにより、スレッドが後述する参照処理の対象とされるまでの間だけ、つまり、スレッドが、そのスレッドに対応するスタック内等に格納されているオブジェクトポインタが指すオブジェクトへのマーク付与に相当する処理の対象とされるまでの間だけ、インタプリタによるそのスレッドの実行時にオブジェクトポインタがスレッドによる処理対象とされたことを検出して作業用メモリ領域、即ちオブジェクト参照情報のメモリ領域に格納する処理を行うため、一旦スレッドが参照処理の対象とされた後は、そのスレッドはインタプリタによる実行時に、その検出がない分だけ、より高速に動作するようになる。

また、前記検査処理は、選択された当該スレッドに対応するスタック内のオブジェクトポインタが指すオブジェクトを、前記アクセス可能であるとして検出した際において、検出した当該オブジェクトが既に非解放対象として管理されておらず、かつ、当該オブジェクト内にオブジェクトポインタがあるときに限り、当該オブジェクトポインタが指すオブジェクトを更に前記アクセス可能であるとして検出する手順を、繰り返し行うことを内容とする処理であり、前記選択手段は、最初の選択後は、前記検査手段により前記検査処理が行われた後において前記複数スレッドのうち前記検査処理を実施されていないスレッドがある限り更なる選択を行い、前記選択手段は、各スレッドに関する情報を参照し、所定のスレッド選定条件に基づき、前記選択を行うこととしてもよい。

これにより、既に非管理対象として管理されたオブジェクトの配下のオブジェクトを重複的に検出のための処理の対象とすることを防止することができる。また、この重複的に検出のための処理を行わない仕組みにより、参照処理の対象として早期に選択されたスレッドより、遅く選択されたスレッドの方が参照処理に係る時間が短縮される可能性が高まるので、各スレッドに要求される応答性能に鑑みて、予めスレッド選定条件を定めておくことで、各スレッドが適切な応答性能を発揮して実行されるようにある程度制御することができるようになる。

また、前記スレッド選定条件は、スレッド状態がwait状態であるスレッドをスレッド状態がwait状態以外であるスレッドよりも早期に選択することを示す条件を含み、前記選択手段は、前記選択を行う時点でwait状態のスレ

ドがある限り、当該wait状態のスレッドを選択することとしてもよい。

これにより、wait状態のスレッドを停止することになるため、現在動作しているAPスレッドに対する悪影響の発生が抑えられる。

また、前記スレッド選定条件は、スレッド優先度の低いスレッドをスレッド優先度の高いスレッドよりも早期に選択することを示す条件を含むこととしてもよい。

これにより、スレッド優先度の高いスレッドは、参照処理が短時間でなされる可能性が高くなり、実行性能の低下がある程度防止されるようになる。

また、前記スレッド選定条件は、スレッドに対応するスタックサイズが小さいスレッドをスタックサイズが大きいスレッドよりも早期に選択することを示す条件を含むこととしてもよい。

これにより、スレッドからアクセス可能なオブジェクトへのオブジェクトポインタを格納するスタックの有効範囲が小さい程、参照処理に要する時間は短くなる傾向にあることに鑑みれば、各スレッドの停止時間のある程度均等化する効果が得られ、特定のスレッドの応答性が他と比べて際立って悪くなるという事態がある程度回避できるようになる。

また、前記ガーベジコレクションシステムは、メモリ管理ユニット（MMU）を用いてメモリを管理するメモリ管理機構を備え、オブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保し、前記解放手段は、前記メモリ管理機構を介して、メモリ領域の解放を行うこととしてもよい。

これにより、MMUを用いるメモリ管理機構により確保されているC言語その他によるプログラムのデータ領域等と同等の位置付けで、オブジェクトに対するメモリ領域が確保されるため、仮に、特定のヒープ領域を独自に管理してそのヒープ領域内の一部をオブジェクトに対するメモリ領域として確保するような制御を行うこととした場合と比べて、オブジェクトの生成前から無用に大きなヒープ領域を確保しておく必要がない点、そのヒープ領域についてのメモリコンパクション処理をする必要がない点等といった有利な効果が得られる。

30 図面の簡単な説明

図1は、本発明の実施形態1に係るGCシステムの機能ブロック図である。

図2は、オブジェクトとオブジェクトポインタとの関係を例示する図である。

図3は、fromテーブル及びtoテーブルを示す図である。

5 図4Aは、参照処理前のfromテーブルの状態を示す図である。

図4Bは、参照処理後のfromテーブル及びtoテーブルの状態を示す図である。

図5は、スレッド情報、スタック及びオブジェクト参照情報を示す図である。

図6は、参照処理とAPスレッドの状態との関係を示す図である。

10 図7は、スレッド選定条件の内容を示す図である。

図8は、GC制御処理を示すフローチャートである。

図9は、対象スレッド決定処理を示すフローチャートである。

図10は、共有オブジェクト参照処理を示すフローチャートである。

図11は、対象スレッド参照処理を示すフローチャートである。

15 図12は、参照処理を示すフローチャートである。

図13は、命令実行処理を示すフローチャートである。

図14は、オブジェクトチェーン追跡処理を示すフローチャートである。

図15は、本発明の実施形態2に係るGCシステムの機能ブロック図である。

図16は、スレッド情報及びスタックを示す図である。

20 図17は、実施形態2におけるGC制御処理を示すフローチャートである。

図18は、実施形態2における命令実行処理を示すフローチャートである。

図19は、従来のJava実行環境により管理されたJavaオブジェクトと、C言語のプログラムにおけるデータとのメモリ配置を示す図である。

25 発明を実施するための最良の形態

<1. 実施形態1>

以下、本発明の実施形態1に係るガーベジコレクション（GC）システムについて図面を用いて説明する。

<1-1. 構成>

30 図1は、本発明の実施形態1に係るGCシステムの機能ブロック図である。

GCシステム10は、CPU、メモリ等を備えたコンピュータにおいてメモリに格納された制御プログラムがCPUによって実行されることにより実現され、マルチスレッド制御を行う一般のオペレーティングシステム（OS：Operating System）及びいわゆる仮想マシンを含み、Java言語等で作成されたアプリケーションプログラムの実行環境として位置づけられるシステムである。

GCシステム10は、図1に示すように、インタプリタ部100、オブジェクト管理部200、スレッド管理部300及びGC部400を備える。

ここで、インタプリタ部100は、基本的にインタプリタであってAPを実行する機能を担い、命令実行部110及びオブジェクト参照検出部120を有する。

オブジェクト管理部200は、オブジェクトを管理する機能を担い、fromテーブル221、toテーブル222等に対応するメモリ領域であるオブジェクト管理情報記憶部210、オブジェクト生成部230及びテーブル切替部240を有する。

なお、fromテーブル221は、GC開始前において存在する全オブジェクトに対する全オブジェクトポインタを格納するものと位置づけられるテーブルであり、toテーブル222は、マークアンドスイープ方式でのマーク付与がなされたオブジェクトに対するオブジェクトポインタを格納するために利用されるテーブルである。これらのテーブルについては後に詳細に説明する。

スレッド管理部300は、マルチスレッド制御を実現し、スレッドを管理する機能を担うものであり、スレッド制御部310と、APスレッド毎に、スレッド情報320、オブジェクト参照情報330及びスタック340に対応するメモリ領域を有する。

なお、スレッド情報320は、GC処理全体の開始時点でONにされそのAPスレッドに対しての参照処理が終了するとOFFにされるGCフラグを含む。参照処理は、オブジェクトへのマーク付与に相当する処理であり、オブジェクトを参照するためにスタック内等に格納されているオブジェクトポインタと同一内容のオブジェクトポインタをfromテーブルからtoテーブルに移動することを主内容とする処理である。

GC部400は、基本的にいわゆるガーベジコレクタに相当し、GC制御部410、スレッド選定条件記憶部420、参照処理部430及び解放部440を有

する。このGC部400を中心として行われるGCは、基本的にマークアンドスイープ方式を用い、参照処理の対象となるAPスレッドを順番に選択して、選択したAPスレッドを停止させて参照処理を行ってからその停止を解除する方式のものである。

- 5 インタプリタ部100における命令実行部110は、APスレッドを構成する命令列を逐次解釈して実行し、命令がオブジェクトの生成命令である場合にはオブジェクト生成部230にオブジェクトを生成させる機能を有する。

- 10 オブジェクト参照検出部120は、GCフラグがONである場合に限り、命令実行部110により実行されるスレッド内の命令が、オブジェクトのオブジェクトポインタを処理対象とする命令であるときに、そのオブジェクトポインタを、オブジェクト参照情報330のメモリ領域の中に格納する機能を有する。

- 15 オブジェクト管理部200におけるオブジェクト生成部230は、いわゆるクラスファイル等のオブジェクトの定義情報を参照して、従来のOSのメモリ管理機構に対して、オブジェクトに必要な量のメモリ領域の確保を要求することを通じて、メモリ内にオブジェクトを生成する機能を有する。なお、GCシステム10は、メモリ管理ユニット（MMU）を利用して物理的なメモリを論理アドレス空間に対応付け、メモリの確保及び解放を管理する従来のOSのメモリ管理機構をも含んでおり、このメモリ管理機構は、ある量のメモリ領域の確保が要求されると、論理アドレス空間内の未使用領域のうち、要求された量のメモリ領域を確保したものとして管理し、その確保したメモリ領域の先頭アドレスを返却する機能を有している。従って、オブジェクト生成部230は、メモリ管理機構を通じて、オブジェクトに対応する論理アドレス空間内の領域を確保することにより、メモリ内にオブジェクトを生成する。
- 20

- 25 なお、この論理アドレス空間には、インタプリタ部100により実行される、Java言語等で作成されたAPにおけるオブジェクト以外に、インタプリタ部100を介さず従来の一般的なOSの直接制御下で実行されるプログラムのデータ、つまり例えばコンパイラ等により既に実行形式にされているプログラムに係るデータ（以下、「Nativeデータ」という。）も配置される。このインタプリタ部100を介さずに実行可能な形式のプログラムが、Nativeデータを配置するために従来のOSのメモリ管理機構を通じてメモリ領域を確保し、そ
- 30

ここにNativeデータを配置するのと同様の仕組みで、オブジェクト生成部230により生成されるオブジェクトについてのメモリ領域の確保はなされる。

かかるNativeデータやそれに係るプログラムは、GCシステム10の特徴となるGCの動作とは直接関係しないので、ここでは詳しい説明を省略する。

- 5 なお、上述したAPスレッドとは、インタプリタ部100を介して実行されるAPの実行単位となるスレッドをいい、つまりNativeデータに係るプログラムの実行単位となるスレッド以外のスレッドをいう。また、ここでいうオブジェクトにはNativeデータは含まれない。

- 10 テーブル切替部240は、fromテーブルを指すものとして用いられているポインタとtoテーブルを指すものとして用いられているポインタとを交換することにより、瞬時にfromテーブルの内容とtoテーブルの内容とを交換したに等しい効果を生じさせる機能を有する。

- 15 スレッド管理部300におけるスレッド制御部310は、マルチスレッド制御を行い、APを構成する各APスレッド及びGC処理のスレッドを含む各スレッドを並列的に実行する。即ち、スレッド制御部310は、微小時間毎に各スレッドを切り替えて実行する。なお、スレッド管理部300は、Nativeデータに係るプログラムのスレッドも並列的に実行するが、このマルチスレッド制御の機能は、基本的に従来のOS等が有する機能であり、GCシステム10の特徴となるGCの動作とは直接関係しないので、ここでは、主に、APスレッド及びGC処理のスレッドにのみ着目して説明する。
- 20

GC部400におけるスレッド選定条件記憶部420は、参照処理を行う対象となるAPスレッドを選択するための条件を示すスレッド選定条件を記憶しているメモリ領域である。

- 25 参照処理部430は、参照処理を行う機能を有し、解放部440は、マーク付与がなされていないオブジェクトの解放、即ち、GCの終了段階においてfromテーブル221に残存しているオブジェクトポインタが指すオブジェクトに対応するメモリ領域を解放する機能を有する。このメモリ領域の解放は、オブジェクトポインタつまりオブジェクトの論理アドレスを指定して、MMUを利用した従来のOSのメモリ管理機構に対する解放要求を行うことにより実現される。なお、GCシステム10に含まれる従来のOSのメモリ管理機構は、論理アドレス
- 30

を指定した解放要求に対して、その論理アドレスと対応付けて管理している確保領域を、新たにメモリを確保する必要がある際に割り当てることができる未使用領域として管理する。

また、GC制御部410は、GC制御処理を実行する機能を有する。即ち、GC制御部410は、スレッド選定条件を参照することにより、参照処理の対象にするAPスレッドを1つ選定し、選択したAPスレッドについて、スレッド制御部310にそのAPスレッドを停止させてから、参照処理部430にそのAPスレッドに対応するスタック及びオブジェクト参照情報に基づいて参照処理を行わせ、その後にスレッド制御部310に選択したAPスレッドの停止を解除させてから、次のAPスレッドの選択を行うという手順を、未処理のAPスレッドが存在しなくなるまで繰り返してから、解放部440に、マーク付与がなされていないオブジェクトの解放を行わせる機能を有する。

<1-2. データ>

以下、GCシステム10において取り扱われるデータについて説明する。

図2は、オブジェクトとオブジェクトポインタとの関係を例示する図である。同図では、オブジェクトポインタを、小さい塗り潰し円で表現している。

オブジェクトを指すつまりメモリ500に対応する論理アドレス空間に配置されたオブジェクトの所在位置を指すオブジェクトポインタは、オブジェクト内、スタック内、又は共有オブジェクト管理情報353内に存在し得る。ここで、共有オブジェクト管理情報353は、APの実行環境としての仮想マシンにおいて必要なため確保されているオブジェクト群に対するオブジェクトポインタ群を含むデータである。

図2の例では、共有オブジェクト管理情報353内のあるオブジェクトポインタがオブジェクト201cを指している。

また、APスレッド351に対応するスタック内のあるオブジェクトポインタは、オブジェクト201aを指しており、つまり、オブジェクト201aのメモリアドレスを内容としており、また別のオブジェクトポインタは、オブジェクト201dを指している。従って、APスレッド351は、実行中にこれらのオブジェクトポインタを参照することによりオブジェクト201a、オブジェクト201dにアクセス可能な状態である。

また、APスレッド352に対応するスタック内のあるオブジェクトポインタは、オブジェクト201bを指しており、オブジェクト201b内のデータとして、オブジェクト201eを指すオブジェクトポインタが含まれている。従って、APスレッド352は、これらのオブジェクトポインタを辿ることによりオブジェクト201eにアクセス可能な状態である。

また、図2は、インタプリタ部100を介して実行されるJava言語等で作成されたAPに対応するオブジェクト201a~201e以外にも、同じくOSのメモリ管理機構により領域が確保されたものであるNativeデータ501a~501dがメモリ500内に混在していることをも表している。即ち、オブジェクト201a~201eは、まとめて、ある種のヒープ領域中に置かれていたのではなく、各Nativeデータと同様に、OSのメモリ管理機構によりそれぞれ個別に領域が割り当てられている。

なお、図2には示さなかったが、スレッド情報320、スタック340及びオブジェクト参照情報330もメモリ500内に確保されるものであり、また、オブジェクト管理情報記憶部210は、メモリ500中の一部に相当する。なお、インタプリタ部100を介して実行されるAPのプログラム部分や、Nativeデータにアクセスするプログラム部分は、読み書き可能なメモリ(RAM)中に置かれるものとしても、読み出しだけ可能なメモリ(ROM)に置かれるものとしてもよい。

図3は、fromテーブル及びtoテーブルを示す図である。

メモリ500内のオブジェクト管理情報記憶部210内に、オブジェクトポインタ又はnull値を十分な数だけ格納するための2つのテーブルが設けられており、また、fromテーブルポインタ211及びtoテーブルポインタ212が存在する。fromテーブルポインタ211は、その2つのテーブルのうち一方を指すポインタであり、toテーブルポインタ212は、他方を指すポインタである。ここでは、現時点でfromテーブルポインタ211が指しているテーブルをfromテーブルと称しており、toテーブルポインタ212が指しているテーブルをtoテーブルと称している。

fromテーブル221は、GC開始時点においては、その時に存在する全オブジェクトに対する全オブジェクトポインタを格納している。

また、t oテーブル222は、参照処理の過程でf r o mテーブル221内から移動してオブジェクトポインタを格納するテーブルである。なお、f r o mテーブル221中のオブジェクトポインタがt oテーブル222へと移動されるときには、そのオブジェクトポインタが格納されていたf r o mテーブル221内の位置にあたるメモリ内容はn u l l値にされる。

図4A及び図4Bは、参照処理によるf r o mテーブル及びt oテーブルの内容の変化を示すための図であり、図4Aは参照処理開始前の状態を示し、図4Bは参照処理後の状態を示す。

図4Aでは、参照処理開始前におけるf r o mテーブル221xは、o b j A 202a、o b j B 202b、o b j C 202cのそれぞれのオブジェクトを指すオブジェクトポインタを含んでいる。なお、図4A及び図4Bでは、各オブジェクトとN a t i v eデータとがメモリ500内に混在していることをも表現している。

その後、APスレッド354を対象として参照処理が行われた場合には、APスレッド354のスタック内に含まれているオブジェクトポインタと同値のものがf r o mテーブルからt oテーブルに移動され、その結果として図4Bに示す状態になる。f r o mテーブル221yでは、o b j A 202aを指していたオブジェクトポインタの存在した場所と、o b j C 202cを指していたオブジェクトポインタの存在した場所との両方がn u l l値に更新されており、t oテーブル222yには、o b j A 202aを指すオブジェクトポインタと、o b j C 202cを指すオブジェクトポインタとが格納されている。

図5は、スレッド情報、スタック及びオブジェクト参照情報を示す図である。

スレッド情報320は、APスレッド毎に、そのスレッド生成時に生成され、そのAPスレッドについての情報を含むものであり、具体的には、状態321、優先度322、スタックポインタ323、GC開始時スタックポインタ324、GCフラグ325、オブジェクト参照情報先頭ポインタ326及びオブジェクト参照情報カレントポインタ327を含む。なお、APスレッド生成時には、スレッド情報320のメモリ領域のみならずスタック340のメモリ領域も確保される。

スレッド情報320における状態321は、マルチスレッド制御のためのスレ

ッド状態を示す情報であり、wait状態、run状態、ready状態等の別を示す。

- 優先度322は、スレッドの優先度を示す情報であり、優先度は例えばスレッド生成時にAPからの指定を受けて定まる。なお、マルチスレッド制御において
- 5 優先度が高いスレッドがready状態であればそれより優先度の低いスレッドよりも優先的にrun状態にされる。

- スタックポインタ323は、該当のスレッドについてのスタック内における現在の有効なデータ範囲の終端を示すものである。なお、マルチスレッド制御において、スレッドがrun状態から、他の状態つまり停止した状態に切り替えられる際に、それまでスタックポインタを指す所定レジスタに格納されていた値がこのスタックポインタ323に格納され、スレッドがrun状態に切り替えられる際に、このスタックポインタ323内の値がその所定レジスタに設定される。
- 10

GC開始時スタックポインタ324は、GC開始時におけるスタック内における有効なデータ範囲の終端を示すものである。

- 15 GCフラグ325は、GC処理全体の開始時点でONにされ、そのAPスレッドに対しての参照処理が終了するとOFFにされる。

オブジェクト参照情報先頭ポインタ326は、該当APスレッドに対応するオブジェクト参照情報のメモリ領域の先頭を示すものであり、APスレッド生成の際のそのメモリ領域確保時に設定される。

- 20 また、オブジェクト参照情報カレントポインタ327は、該当APスレッドに対応するオブジェクト参照情報のメモリ領域において、オブジェクト参照検出部120が次にオブジェクトポインタを格納すべき位置を示す情報であり、オブジェクト参照検出部120によって参照及び更新される。

図6は、参照処理とAPスレッドの状態との関係を示す図である。

- 25 GCスレッド356は、GC制御部410によるGCを実行するスレッドであり、GCとして各APスレッドを順番に対象として参照処理を行う。ここでは、APスレッド355a、APスレッド355b、APスレッド355cの順に参照処理が行われる場合の例を示している。

APスレッド355aは、参照処理がなされた後の状態であり、実行中である。

- 30 ここでいう実行中は、sleep状態つまり停止状態ではないことであり、各瞬

時においてはrun状態やready状態等と変化し得る。

APスレッド355bは、参照処理がなされているところの状態であり、停止している。参照処理は、APスレッドを強制的に停止状態にして、スタック及びオブジェクト参照情報を参照して行われる。

- 5 APスレッド355cは、参照処理が未だなされていない状態であり、実行中である。

図7は、スレッド選定条件の内容を示す図である。

- 10 スレッド選定条件421は、どのAPスレッドを優先して参照処理の対象とするかの判断基準となる情報であり、スレッド状態422、スレッド優先度423及びスタックサイズ424から構成される。

スレッド状態422は、どの状態のAPスレッドを優先するかを示す情報であり、図7の例はwait状態のAPスレッドを、run状態等のAPスレッドより優先する旨を示している。

- 15 スレッド優先度423は、APスレッドについて定められている優先度の高いものを参照処理の対象として優先的に選定するか、又は低いものを優先的に選定するかを示す情報であり、図7の例では優先度の低いものを優先的に選定する旨を示している。

- 20 また、スタックサイズ424は、APスレッドに対応するスタック領域の有効範囲の大きさが、大きいところのAPスレッドを優先的に選定するか、又は小さいところのAPスレッドを優先的に選定するかを示す情報であり、図7の例ではスタック領域の有効範囲の大きさが小さいものを優先的に選定する旨を示している。

<1-3. 動作>

以下、GCシステム10の動作について説明する。

- 25 図8は、GC制御処理を示すフローチャートである。

GC制御処理は、タイマーに基づき一定周期で行われる。

- 30 GC制御部410は、最初に、fromテーブルポインタ211とtoテーブルポインタ212の内容を交換することにより、fromテーブルとtoテーブルとの内容を切り替える（ステップS11）。これにより、現在解放されていない全てのオブジェクトを指す全てのオブジェクトポインタがfromテーブルに

格納されている状態になる。

続いて、GC制御部410は、スレッド制御部310に全APスレッドを停止させ（ステップS12）、全APスレッドについてのスレッド情報中のGCフラグ325をONにし、現在のスタックポインタをスレッド情報中のGC開始時スタックポインタ324に設定し（ステップS13）、スレッド制御部310に全APスレッドの停止を解除させる（ステップS14）。ステップS13においては、更に、APスレッド毎に、オブジェクト参照情報のメモリ領域を確保し、そのメモリ領域の先頭を指すポインタをオブジェクト参照情報先頭ポインタ326及びオブジェクト参照情報カレントポインタ327に設定する。

- 10 なお、スレッド制御部310は、従来のOSのマルチスレッド制御機構と同様の方式によりスレッドの停止及び停止解除を行う。このスレッドの停止は、スレッドを停止状態、即ちsleep状態にする制御であり、スレッドの停止解除とは、sleep状態を解除して、ready状態に戻す制御である。

- 15 ステップS14に続いて、GC制御部410は、共有オブジェクト管理情報353内のオブジェクトポインタを辿ることで到達するオブジェクトに対するマーク付与を行うための共有オブジェクト参照処理を行い（ステップS15）、GC未処理スレッドつまり未だ参照処理の対象として選定されていないAPスレッドが存在するか否かを判定する（ステップS16）。なお、共有オブジェクト参照処理については後述する。

- 20 ステップS16においてGC未処理スレッドが存在すると判定した場合には、GC制御部410は、参照処理の対象となるAPスレッドを決定するための対象スレッド決定処理を行い（ステップS17）、スレッド制御部310にその決定したAPスレッドを停止させ（ステップS18）、その決定したAPスレッドに関しての参照処理を内容とする対象スレッド参照処理を実行し（ステップS19）、
- 25 その決定したAPスレッドのGCフラグ325をOFFにし（ステップS20）、スレッド制御部310にその決定したAPスレッドの停止を解除させて（ステップS21）、再度、ステップS16の判定に戻る。なお、対象スレッド決定処理及び対象スレッド参照処理については、後述する。また、ステップS20の直後に、GC制御部410は、対象のAPスレッドに対応するオブジェクト参照情報のメモリ領域を解放する。
- 30

また、ステップS 1 6において、GC未処理スレッドが存在しないと判定した場合には、GC制御部4 1 0は、解放部4 4 0にマーク付与がなされていないオブジェクトを解放させ（ステップS 2 2）、GC制御処理を終える。ステップS 2 2においては、解放部4 4 0は、参照処理によってtoテーブルにオブジェクト
5 トポインタが移行されていないところのオブジェクト、つまりfromテーブルに残存しているオブジェクトポインタが指すオブジェクトに対応するメモリ領域を解放する。この解放は、オブジェクト生成部2 3 0が、オブジェクト生成のためにオブジェクトに割り当てるためのメモリ領域を、MMUを利用したOSのメモリ管理機構に基づき確保することに対応し、その確保されたメモリ領域の解放
10 を意味する。なお、この解放されたメモリ領域は、メモリ管理機構により、新たにオブジェクトやNativeデータの格納域として割り当てられ得るようになる。

図9は、対象スレッド決定処理を示すフローチャートである。

GC制御部4 1 0は、スレッド選定条件記憶部4 2 0内のスレッド選定条件4
15 2 1を参照して、対象スレッド決定処理を行う。

まず、GC制御部4 1 0は、各APスレッドに対応するスレッド情報を参照して、状態3 2 1がwait状態のAPスレッドを検索する（ステップS 3 1）。この検索結果のスレッド数を判定し（ステップS 3 2）、1であれば、その検索結果のAPスレッドを参照処理の対象スレッドとして決定し（ステップS 3 7）、
20 対象スレッド決定処理を終える。

また、ステップS 3 2において検索結果のスレッド数が0であれば、GC制御部4 1 0は、スレッドの優先度3 2 2が最も低いAPスレッドを検索し（ステップS 3 3）、検索結果のスレッド数は1か否かを判定する（ステップS 3 5）。また、ステップS 3 2において検索結果のスレッド数が2以上であれば、検索結果のAPスレッドの中から更に絞り込むため、スレッドの優先度3 2 2が最も低いAPスレッドを検索し（ステップS 3 4）、検索結果のスレッド数は1か否かを判定する（ステップS 3 5）。
25

ステップS 3 5においてスレッド数が1と判定した場合には、GC制御部4 1 0は、その検索結果のAPスレッドを参照処理の対象スレッドとして決定し（ステップS 3 7）、対象スレッド決定処理を終える。
30

また、ステップS35において、検索結果のスレッド数が1でないと判定した場合、即ち、最も低い優先度のAPスレッドが複数存在した場合には、GC制御部410は、検索結果のAPスレッドの中からスタックサイズが最小のAPスレッドを検索し（ステップS36）、スタックサイズが最小のAPスレッドを参照
5 処理の対象スレッドとして決定し（ステップS37）、対象スレッド決定処理を終える。

図10は、共有オブジェクト参照処理を示すフローチャートである。

GC制御部410は、オブジェクトポインタ群を含む共有オブジェクト管理情報353の先頭のオブジェクトポインタに着目して（ステップS41）、参照処
10 理部430に参照処理（ステップS42）を行わせることにより、APの実行環境としての仮想マシンにおいて必要なため確保されているオブジェクト群全てにマーク付与を行う。

図11は、対象スレッド参照処理を示すフローチャートである。

GC制御部410は、対象スレッドに対応するスレッド情報中のGC開始時スタックポインタ324が指す位置に着目し（ステップS51）、参照処理部43
15 0に参照処理（ステップS52）を行わせることにより、スタック領域内のオブジェクトポインタから辿ることのできる全てのオブジェクトへのマーク付与を行う。なお、GC制御処理（図8参照）の開始時点と、この対象スレッド参照処理を開始する時点は異なるため、GC開始時スタックポインタ324が指す位置と
20 それに続く各位置の内容は、GC開始時点とは異なっているが、後続するステップS53及びステップS54によって、GC開始以後に、対象スレッドの動作によって変化したオブジェクトポインタから辿ることのできる全てのオブジェクトへのマーク付与が行われるので、過剰にマーク付与をする場合はあり得るが、参照されているにも関わらずマーク付与が漏れるという事態は生じない。

25 続いて、GC制御部410は、オブジェクト参照情報先頭ポインタ326の指す位置に着目し（ステップS53）、参照処理部430に参照処理（ステップS54）を行わせ、対象スレッド参照処理を終える。

図12は、参照処理を示すフローチャートである。

参照処理部430は、着目されているメモリ位置から、オブジェクトポインタ
30 を検索し（ステップS61）、オブジェクトポインタを検出したか否かを判定し

(ステップS62)、検出した場合にはその検出したオブジェクトポイントと同一のオブジェクトポイントがfromテーブルに存在するか否かを判定し(ステップS64)、fromテーブルに存在すれば、fromテーブルからtoテーブルにそのオブジェクトポイントをコピーして、fromテーブル内のそのオブジェクトポイントが存在していた位置にnull値を記録し(ステップS66)、そのオブジェクトポイントが指すオブジェクトのデータ先頭に着目し(ステップS67)、再度ステップS61に戻る。

また、ステップS64において、fromテーブルにはそのオブジェクトポイントが存在しないと判定した場合には、既にfromテーブルからtoテーブルに移されているため、参照処理部430は、着目している位置の次の位置に着目し(ステップS65)、再度ステップS61に戻る。ステップS65及びステップS61の組み合わせにより、共有オブジェクト管理情報353内のオブジェクトポイントを次々と検索することや、スタック内のオブジェクトポイントを次々と検索することや、オブジェクト参照情報内のオブジェクトポイントを次々と検索することや、あるオブジェクト内のデータメンバとしてのオブジェクトポイントを次々と検索すること等が実現される。

また、ステップS62においてオブジェクトポイントを検出できなかったと判定した場合には、参照処理部430は、着目位置がオブジェクト内であるか否かを判定する(ステップS63)。なお、オブジェクトポイントを検出できなかったというのは、着目位置が共有オブジェクト管理情報353内であればその共有オブジェクト管理情報353内においてオブジェクトポイントを検出できなかったということであり、着目位置がスタック内であればスタック内においてオブジェクトポイントを検出できなかったということであり、着目位置がオブジェクト参照情報内であればそのオブジェクト参照情報内においてオブジェクトポイントを検出できなかったということであり、着目位置がオブジェクト内であればそのオブジェクト内においてオブジェクトポイントを検出できなかったということである。

ステップS63において着目位置がオブジェクト内であると判定した場合には、参照処理部430は、そのオブジェクト内に着目前の着目位置の次位置に着目し(ステップS68)、再度ステップS61に戻る。このステップS68によって、

参照処理部430は、ステップS67によってオブジェクト内に着目する前に着目していたオブジェクトポインタの次の位置に着目することになる。

また、ステップS63において着目位置がオブジェクト内ではないと判定した場合には、参照処理部430は、参照処理を終える。

5 図13は、命令実行処理を示すフローチャートである。

インタプリタ部100における命令実行部110は、run状態にされたAPスレッドにおけるプログラム中の命令記述を逐次解釈して実行する命令実行処理を行う。

まず、命令実行部110は、APスレッドの現在の実行位置における命令を解釈して実行し（ステップS71）、オブジェクト参照検出部120は、そのAPスレッドに対応するスレッド情報中のGCフラグ325がONであるか否かを判定し（ステップS72）、ONでなければ、命令実行部110が次の命令の解釈実行を行う（ステップS71）。

15 ステップS72において、GCフラグ325がONであると判定した場合には、オブジェクト参照検出部120は、ステップS71において命令実行部110により実行された命令が、オブジェクトポインタを処理対象とする命令であったか否かを判定し（ステップS73）、オブジェクトポインタを処理対象とする命令でなかったならば、命令実行部110が次の命令の解釈実行を行う（ステップS71）。なお、オブジェクトポインタを処理対象とする命令とは、スタック内の
20 オブジェクトポインタをオブジェクト内にコピーする演算を指示内容とする命令や、オブジェクト内のオブジェクトポインタを他のオブジェクト内にコピーする演算を指示内容とする命令である。

ステップS73において、オブジェクトポインタを処理対象とする命令であったと判定した場合には、オブジェクト参照検出部120はそのオブジェクトポインタと同値のものがオブジェクト参照情報領域に既に格納済みであるか否かを判定し（ステップS74）、格納済みである場合には、命令実行部110が次の命令の解釈実行を行う（ステップS71）。

ステップS74において、そのオブジェクトポインタと同値のものがオブジェクト参照情報のメモリ領域に格納済みでない場合には、オブジェクト参照検出部
30 120は、そのオブジェクトポインタを、オブジェクト参照情報カレントポイン

タ 3 2 7 が指すオブジェクト参照情報中の位置に格納し（ステップ S 7 5）、そのオブジェクトポインタの指すオブジェクトに着目して、オブジェクトチェーン追跡処理（ステップ S 7 6）を行い、その後は命令実行部 1 1 0 が次の命令の解釈実行を行う（ステップ S 7 1）。

- 5 なお、ステップ S 7 1 における実行対象とされる命令がオブジェクトを生成する命令である場合には、命令実行部 1 1 0 は、オブジェクト生成部 2 3 0 にオブジェクトの生成を指示し、これを受けてオブジェクト生成部 2 3 0 は、オブジェクトを生成するとともにそのオブジェクトを指すオブジェクトポインタを AP スレッドに対して返却し、そのオブジェクトポインタのコピーをメモテーブル内に
- 10 設定する。

図 1 4 は、オブジェクトチェーン追跡処理を示すフローチャートである。

- オブジェクト参照検出部 1 2 0 は、着目しているオブジェクトに未着目のオブジェクトポインタが存在するか否かを判定し（ステップ S 8 1）、存在する場合には、その未着目の 1 つのオブジェクトポインタに着目し（ステップ S 8 2）、
- 15 その着目したオブジェクトポインタをオブジェクト参照情報のメモリ領域に格納し（ステップ S 8 3）、その着目したオブジェクトポインタの指すオブジェクトに着目して更にオブジェクトチェーン処理（ステップ S 8 1 ～ S 8 4）を行う（ステップ S 8 4）。なお、オブジェクトポインタをオブジェクト参照情報のメモリ領域に格納したときには、オブジェクト参照検出部 1 2 0 は、オブジェクト参照
- 20 情報カレントポインタをそのオブジェクトポインタのサイズ分だけ進める。

また、ステップ S 8 1 において、着目しているオブジェクトに未着目のオブジェクトポインタが存在していないと判定した場合には、オブジェクト参照検出部 1 2 0 は、オブジェクトチェーン追跡処理を終える。

- 従って、この命令実行処理及びオブジェクトチェーン追跡処理によって、GC
- 25 フラグ 3 2 5 が ON である場合において、オブジェクトポインタが処理対象とされたときにはそのオブジェクトポインタから辿ることのできるオブジェクトポインタはいずれもオブジェクト参照情報のメモリ領域に格納されることになる。

- なお、AP スレッドによるオブジェクトポインタを処理対象とする演算の実行によって、そのオブジェクトポインタで指されていたオブジェクトにその AP ス
- 30 レッドからアクセスすることが不可能な状態となった場合、つまり、AP スレ

5 ドの動作により、(a) オブジェクトポインタがクリアされた場合、(b) オブジェクトポインタが他の内容に更新された場合、或いは、(c) オブジェクトポインタがスタック領域の有効範囲内に存在する場合においてスタックポインタの変更がなされてスタック領域の有効範囲外となりAPスレッドから基本的にアクセス不能となった場合にも、そのAPスレッドによって予めそのオブジェクトポインタが他のAPスレッドからはアクセス可能な場所にコピーされていたならば、そのオブジェクトポインタで指されるオブジェクトにはマーク付与が必要となるので、そのために、上述したステップS72～S76、ステップS81～S84の処理が行われる。

10 <2. 実施形態2>

以下、本発明の実施形態2に係るGCシステムについて図面を用いて説明する。

15 実施形態2に係るGCシステムは、APスレッドを順次停止して参照処理を行う点で、実施形態1に係るGCシステム10と基本的に同様である。ただし、実施形態2に係るGCシステムは、主に、APスレッド毎にオブジェクト参照情報やGCフラグを有するのではなくシステム全体でまとめてオブジェクト参照情報やGCフラグを有する点や、スレッド情報の内容を一般的なマルチスレッド制御に要する程度に縮小している点が、GCシステム10と異なる。

図15は、本発明の実施形態2に係るGCシステムの機能ブロック図である。

20 図15に示すGCシステム20の構成要素のうち、実施形態1で示したGCシステム10と同様の構成要素については、図1と同じ符号を付しており、ここではGCシステム20に特有の点を中心に説明する。なお、特に説明しない点については、GCシステム20はGCシステム10と同様である。

GCシステム20は、図15に示すように、インタプリタ部1100、オブジェクト管理部200、スレッド管理部1300及びGC部1400を備える。

25 ここで、インタプリタ部1100は、基本的にインタプリタであってAPを実行する機能を担い、命令実行部110及びオブジェクト参照検出部1120を有する。

30 スレッド管理部1300は、マルチスレッド制御を行いスレッドを管理する機能を担い、スレッド制御部310と、APスレッド毎に、スレッド情報1320及びスタック340に対応するメモリ領域を有する。なお、スレッド情報132

0は、図16に示すように状態321、優先度322及びスタックポインタ323を含むものである。

GC部1400は、基本的にいわゆるガーベジコレクタに相当し、GC制御部1410と、スレッド選定条件記憶部420と、参照処理部1430と、解放部440と、オブジェクト参照情報1450及びGCフラグ1460に対応するメモリ領域とを有する。

インタプリタ部1100におけるオブジェクト参照検出部120は、GCフラグ1460がONである場合に限って、命令実行部110により実行されるAPスレッド内の命令が、オブジェクトのオブジェクトポインタを処理対象とする命令であるときに、そのオブジェクトポインタを、オブジェクト参照情報1450のメモリ領域の中に格納する機能を有する。

GC部1400を中心として行われるGCは、基本的にマークアンドスイープ方式を用い、参照処理の対象となるAPスレッドを順番に選択して、選択したAPスレッドを停止させて参照処理を行ってからその停止を解除する方式のものである。

GC部1400における参照処理部1430は、スタック及びオブジェクト参照情報1450を参照することにより参照処理を行う機能を有する。オブジェクト参照情報1450は、GCシステム10におけるオブジェクト参照情報と同様にオブジェクトポインタを内容とする情報であるが、APスレッド毎に存在するのではない。このオブジェクト参照情報1450のメモリ領域には、全APスレッドについてのオブジェクト参照検出部1120により、オブジェクトポインタが命令の処理対象とされた場合にそのオブジェクトポインタが格納される。なお、スレッド情報1320、スタック340、オブジェクト参照情報1450、GCフラグ1460は、オブジェクトやNativeデータと同様に、オブジェクト管理情報記憶部210を含むメモリに格納される。

また、GC制御部1410は、図17に示すGC制御処理を実行する機能を有する。

図17は、実施形態2におけるGC制御処理を示すフローチャートである。なお、このフローチャート中の処理ステップのうち、実施形態1のGC制御処理と同様のものについては、図8と同じ符号を付して示している。

このGC制御処理は、タイマーに基づき一定周期で行われる。

GC制御部1410は、最初に、fromテーブルポインタ211とtoテーブルポインタ212の内容を交換することにより、fromテーブルとtoテーブルとの内容を切り替え（ステップS11）、GCフラグ1460をONにする
5 （ステップS111）。

ステップS111に続いて、GC制御部1410は、共有オブジェクト管理情報353内のオブジェクトポインタを辿ることで到達するオブジェクトに対するマーク付与を行うための共有オブジェクト参照処理を行い（ステップS15）、GC未処理スレッドつまり未だ参照処理の対象として選定されていないAPスレッドが存在するか否かを判定する（ステップS16）。
10

ステップS16においてGC未処理スレッドが存在すると判定した場合には、GC制御部1410は、参照処理の対象となるAPスレッドを決定するための対象スレッド決定処理を行い（ステップS17）、スレッド制御部310にその決定したAPスレッドを停止させ（ステップS18）、その決定したAPスレッド
15 に対応するスタックポインタの位置に着目して（ステップS112）、参照処理部1430に参照処理（図12参照）を実行させ（ステップS113）、スレッド制御部310にその決定したAPスレッドの停止を解除させて（ステップS21）、再度、ステップS16の判定に戻る。

また、ステップS16において、GC未処理スレッドが存在しないと判定した
20 場合には、GC制御部1410は、スレッド制御部310に全APスレッドを停止させ（ステップS114）、オブジェクト参照情報1450のメモリ領域の先頭位置に着目し（ステップS115）、参照処理部1430に参照処理を実行させ（ステップS116）、GCフラグ1460をOFFにし（ステップS117）、スレッド制御部310に全APスレッドの停止を解除させ（ステップS118）、
25 解放部440にマーク付与がなされていないオブジェクトを解放させ（ステップS22）、GC制御処理を終える。

ここで、インタプリタ部1100によりなされる命令実行処理について簡単に説明する。

図18は、実施形態2における命令実行処理を示すフローチャートである。

30 同図に示すように、この命令実行処理は、図13で示した命令実行処理からス

ステップS76を削除したものと基本的に等しい。

- まず、命令実行部110は、APスレッドの現在の実行位置における命令を解釈して実行し（ステップS71）、オブジェクト参照検出部1120は、GCフラグ1460がONであるか否かを判定し（ステップS72）、ONでなければ、
- 5 命令実行部110が次の命令の解釈実行を行う（ステップS71）。

- ステップS72において、GCフラグ325がONであると判定した場合には、オブジェクト参照検出部1120は、ステップS71において命令実行部110により実行された命令が、オブジェクトポインタを処理対象とする命令であったか否かを判定し（ステップS73）、オブジェクトポインタを処理対象とする命令でなかったならば、命令実行部110が次の命令の解釈実行を行う（ステップ
- 10 S71）。

- ステップS73において、オブジェクトポインタを処理対象とする命令であったと判定した場合には、オブジェクト参照検出部1120はそのオブジェクトポインタと同値のものがオブジェクト参照情報領域に既に格納済みであるか否かを判定し（ステップS74）、格納済みである場合には、命令実行部110が次の命令の解釈実行を行う（ステップS71）。
- 15

- ステップS74において、そのオブジェクトポインタと同値のものがオブジェクト参照情報のメモリ領域に格納済みでない場合には、オブジェクト参照検出部1120は、そのオブジェクトポインタを、オブジェクト参照情報中の位置に格納し（ステップS75）、その後は命令実行部110が次の命令の解釈実行を行う（ステップS71）。
- 20

- このようにしてGC制御部1410は、スレッド選定条件を参照することにより、参照処理の対象にするAPスレッドを1つ選定し、選択したAPスレッドについて、スレッド制御部310にそのAPスレッドを停止させてから、参照処理部1430にそのAPスレッドに対応するスタックに基づいて参照処理を行わせ、その後スレッド制御部310に選択したAPスレッドの停止を解除させてから、次のAPスレッドの選択を行うという手順を、未処理のAPスレッドが存在しなくなるまで繰り返し、その後、スレッド制御部310に全APスレッドを停止させ、参照処理部1430にオブジェクト参照情報に基づいて参照処理を行わせて、
- 25
- 30 スレッド制御部310に全APスレッドの停止を解除させて、解放部440に、

マーク付与がなされていないオブジェクトの解放を行わせる。

<3. 考察>

以下、上述の本発明に係るGCシステムが管理するオブジェクトと、従来のJ
a v a実行環境におけるガーベジコレクタ等により扱われるオブジェクト(以下、
5 特に「J a v aオブジェクト」という。)とのメモリ領域の差異について簡単に
対比説明を行う。

上述の実施形態1、2に示した本発明に係るGCシステムでは、プログラマに
確保したメモリの解放を意識させないようなJ a v a言語等のオブジェクト指向
言語により作成されたA Pを実行する際におけるオブジェクトの生成時のメモリ
10 確保と、GC処理におけるそのオブジェクトに対応するメモリの解放とを、MM
Uを利用した従来のO Sのメモリ管理機構を通じて行う。これにより、オブジェ
クトに対して確保されるメモリ領域は、N a t i v eデータ、つまり他の言語、
例えばC言語等、により作成されたプログラムのデータに対して確保されるメモ
リ領域と、論理アドレス空間において混在する。図3では、この混在するイメー
15 ジを表現している。

これに対して、従来のJ a v a実行環境により管理されたJ a v aオブジェク
トと、ここではC言語のプログラムにおけるデータ(以下、「Cデータ」という。)
とのメモリにおける存在状況を示した図が図19である。

図19に示すように、各C言語のプログラム953、954は、Cデータ92
20 1、922のためのメモリ領域をO Sのメモリ管理機構を通じて、ランダムアク
セスメモリ(RAM)900内に確保する。

また、各J a v aプログラム951、952から生成される各J a v aオブジ
ェクト911~913は、1つのヒープ領域910内に確保され、そのヒープ領
域910内での配置の管理は、O Sによってではなく、ガーベジコレクタ950
25 を含むJ a v a実行環境によって行われる。このため、J a v a実行環境は、ま
ずO Sのメモリ管理機構を通じてヒープ領域を確保しておいて、オブジェクトを
生成する際には、そのヒープ領域内の一領域をそのオブジェクトに対して割り当
てる。このヒープ領域は、他の言語のプログラムが利用できるメモリ量を減らし
てしまう。

30 また、J a v a実行環境は、オブジェクトを解放する場合には、ヒープ領域内

のそのオブジェクトに割り当てられていた領域を、新たにオブジェクトに対して割り当てられ得る未使用領域として管理するが、ヒープ領域内において断片化した全未使用領域を連続させるためのいわゆるメモリコンパクション処理を随時行う必要がある。

- 5 なお、この点、実施形態 1、2 に示した本発明に係る GC システムでは、ヒープ領域の確保及びその内部の管理を行うのではなく、オブジェクトに対するメモリ割り当てを直接的に、MMU を利用した従来の OS のメモリ管理機構を通じて行っているため、当然ながらヒープ領域内のメモリコンパクション等は不要となる。

10 <4. 補足>

以上、本発明に係る GC システムについて、実施形態 1、2 に基づいて説明したが、本発明は、勿論これらの実施形態に限られない。以下、実施形態の変形に関して説明する。

- 15 (1) スレッド選定条件は、スレッド状態、スレッド優先度、スタックサイズという 3 つの条件をこの順に優先適用されることとしたが、条件数や適用順序等は、これに限定されることはなく、また、例えばスレッド優先度が高いものを条件にしたりスタックサイズが大きいものを条件にしたりしてもよい。

- 20 但し、実施形態において示したようにスレッド状態が wait 状態の AP スレッドを参照処理対象と選定することは、現在動作している AP に対する悪影響が
25 少ないという効果につながる。また、参照処理 (図 12) のアルゴリズムにより、ある AP スレッドに対する参照処理にかかる時間は、参照処理を先に行う AP スレッドより後に行う AP スレッドの方が比較的短くなるため、実施形態において示したようにスレッド優先度が低いものを早めに参照処理対象と選定するようにしておくことで、即応性の求められるところのスレッド優先度の高い AP スレ
30 ドの停止時間を短くし得るという効果が生じる。また、同様の理由と、スタックサイズが大きいほど参照処理には時間がかかるという理由とから、対応するスタックサイズの小さい AP スレッドから順に参照処理を行うようにすることで、同じスレッド優先度の AP スレッド間である程度等しくなるように AP スレッドの停止時間を分散できるという効果が生じる。

- 30 (2) 実施形態では、GC 制御処理がタイマーに基づき一定周期で行われること

としたが、これに限定されることはなく、APに割り当てられる空きメモリの量が所定量より少なくなった場合にGC制御処理を行うようにしてもよい。

(3) 実施形態で示したオブジェクト参照情報のメモリ領域は、連続した物理アドレスで表されるメモリ領域であってもよいし、オブジェクトポインタの格納に際して必要が生じた度に一定量のメモリ領域を追加的に確保してその確保した複数の断続的なメモリ領域を論理アドレス上で連続しているように扱うこととしてもよい。

(4) 実施形態1で示したGCシステムにおいては、インタプリタによる命令実行処理中に、GCフラグがONであればオブジェクトポインタを処理対象としたときにオブジェクト参照情報のメモリ領域にそのオブジェクトポインタを保存する処理を行っておくこととし、命令実行処理中では、そのオブジェクトポインタと同値のものをfromテーブルからtoテーブルに移行することを内容とする参照処理を行わずインタプリタによる命令実行の高速性を保つようにしたが、命令実行処理中にその参照処理を行うこととしてもよい。

(5) 実施形態では共有オブジェクト管理情報内のオブジェクトポインタから辿ることのできるオブジェクトへのポインタは参照処理によってfromテーブルからtoテーブルに移行されることとしたが、参照処理を行わなくても、共有オブジェクト管理情報によって管理されているオブジェクトは解放部による解放対象から除外しさえすればよい。

(6) 実施形態で示したGCシステムは、コンピュータ上に実装されるのみならず、CPUを備える携帯端末、家電機器等において実装されることとしてもよい。

(7) 実施形態で示したスレッド制御部は、微小時間毎に各スレッドを切り替えて実行することにより各スレッドを並列的に実行、つまり擬似的に並列実行することとしたが、各スレッドをマルチプロセッサにおける複数のプロセッサエレメントによって実際に並列実行することとしてもよい。

(8) 実施形態では、オブジェクト生成部により生成される各オブジェクトは、従来のOSのメモリ管理機構により、各Nativeデータと同等に、論理アドレス空間にメモリが割り当てられるものであることとしたが、各オブジェクトは、論理アドレス空間におけるひとかたまりのヒープ領域内に、それぞれメモリが割り当てられるように、特別の管理を行うこととしてもよい。この特別の管理は、

ヒープ領域を、従来のOSのメモリ管理機構により確保し、オブジェクトの生成に対応してその内部の未割当領域を各オブジェクトに割り当て、オブジェクトの解放に対応してその内部の領域を未割当領域と定めることにより実現される。この特別の管理をする場合において、更に、ヒープ領域内において断片化した全未割当領域を連続させるためのいわゆるメモリコンパクション処理を随時行う必要がある。なお、このことから、ヒープ領域を利用する方式は、上述の<3. 考察>において示したように必ずしも最適なものではない。

(9) 実施形態で示したGCシステムは従来のOSのメモリ管理機構を含んでいることとしたが、GCシステムの外部に、そのGCシステムの基盤環境として、MMUを用いてメモリを管理する従来のOSのメモリ管理機構が存在することとし、GCシステムはメモリの確保及び解放をその外部のメモリ管理機構を通じて行うこととしてもよい。

(10) 実施形態で示したGCシステムの各機能を実現させるための各種処理(図8～図14、図17、図18参照)をCPUに実行させるためのプログラムを、記録媒体に記録し又は各種通信路等を介して、流通させ頒布することもできる。このような記録媒体には、ICカード、光ディスク、フレキシブルディスク、ROM等がある。流通、頒布されたプログラムは、CPUを備える装置におけるCPUで読み取り可能なメモリ等に格納されることにより利用に供され、そのCPUがそのプログラムを実行することにより実施形態で示したGCシステムの各機能が実現される。

産業上の利用可能性

本発明に係るGCシステムは、アプリケーションプログラム(AP: Application Program)の作成者にメモリ領域の確保や解放を意識させないJava等のオブジェクト指向のプログラミング言語で作成されたAPのためのコンピュータ上の実行環境として利用される。

請 求 の 範 囲

1. 複数スレッドで構成されるオブジェクト指向プログラムの実行過程において不要になったオブジェクトに対応するメモリ領域を解放するガーベジコレクションシステムであって、

5 複数スレッドそれぞれを順に選択する選択手段と、

選択されたスレッドについて、当該スレッドの実行を停止して、当該スレッドからオブジェクトポインタの参照を通じてアクセス可能であるオブジェクトを検出して当該検出したオブジェクトを非解放対象として管理して、当該スレッドの実行を再開するという手順からなる検査処理を実施する検査手段と、

10 前記選択手段による前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象として管理する検知手段と、

15 前記複数スレッド全てについて前記検査処理が完了した後において、非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を、解放する解放手段とを備える

ことを特徴とするガーベジコレクションシステム。

2. 前記検知手段は、前記検知を、実行中のスレッドについて未だ検査処理がなされていない場合に限り行い、

20 前記検知手段は、

前記検知をした場合に、当該実行中のスレッドに処理対象にされたオブジェクトポインタ、及び当該オブジェクトポインタから辿ることのできるオブジェクト内のオブジェクトポインタを、当該スレッドに対応する作業用メモリ領域に保存する検出部と、

25 前記検査手段によりスレッドの実行が停止されている間に、当該スレッドに対応する作業用メモリ領域内のオブジェクトポインタから辿ることのできるオブジェクトを非解放対象として管理する管理部とを有する

ことを特徴とする請求の範囲第1項記載のガーベジコレクションシステム。

3. 前記検査処理は、

30 選択された当該スレッドに対応するスタック内のオブジェクトポインタが指す

オブジェクトを、前記アクセス可能であるとして検出した際において、

検出した当該オブジェクトが既に非解放対象として管理されておらず、かつ、当該オブジェクト内にオブジェクトポインタがあるときに限り、当該オブジェクトポインタが指すオブジェクトを更に前記アクセス可能であるとして検出する手順を、

繰返し行うことを内容とする処理であり、

前記選択手段は、最初の選択後は、前記検査手段により前記検査処理が行われた後において前記複数スレッドのうち前記検査処理を実施されていないスレッドがある限り更なる選択を行い、

10 前記選択手段は、各スレッドに関する情報を参照し、所定のスレッド選定条件に基づき、前記選択を行う

ことを特徴とする請求の範囲第2項記載のガーベジコレクションシステム。

4. 前記スレッド選定条件は、スレッド状態がwait状態であるスレッドをスレッド状態がwait状態以外であるスレッドよりも早期に選択することを示す条件を含み、

15 前記選択手段は、前記選択を行う時点でwait状態のスレッドがある限り、当該wait状態のスレッドを選択する

ことを特徴とする請求の範囲第3項記載のガーベジコレクションシステム。

5. 前記スレッド選定条件は、スレッド優先度のスレッド優先度の低いスレッドをスレッド優先度の高いスレッドよりも早期に選択することを示す条件を含む

20 ことを特徴とする請求の範囲第4項記載のガーベジコレクションシステム。

6. 前記スレッド選定条件は、スレッドに対応するスタックサイズが小さいスレッドをスタックサイズが大きいスレッドよりも早期に選択することを示す条件を含む

25 ことを特徴とする請求の範囲第5項記載のガーベジコレクションシステム。

7. 前記ガーベジコレクションシステムは、メモリ管理ユニット(MMU)を用いてメモリを管理するメモリ管理機構を備え、オブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保し、

30 前記解放手段は、前記メモリ管理機構を介して、メモリ領域の解放を行う

ことを特徴とする請求の範囲第6項記載のガーベジコレクションシステム。

8. 前記スレッド選定条件は、スレッドに対応するスタックサイズが小さいスレッドをスタックサイズが大きいスレッドよりも早期に選択することを示す条件を含む

5 ことを特徴とする請求の範囲第3項記載のガーベジコレクションシステム。

9. 前記スレッド選定条件は、スレッド優先度の低いスレッドをスレッド優先度の高いスレッドよりも早期に選択することを示す条件を含む

ことを特徴とする請求の範囲第3項記載のガーベジコレクションシステム。

10. 前記ガーベジコレクションシステムは、メモリ管理ユニット（MMU）を用いてメモリを管理するメモリ管理機構を利用するものであり、オブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保し、

前記解放手段は、前記メモリ管理機構を介して、メモリ領域の解放を行う

ことを特徴とする請求の範囲第1項記載のガーベジコレクションシステム。

15 11. コンピュータ上において複数スレッドで構成されるオブジェクト指向プログラムの実行過程において不要になったオブジェクトに対応するメモリ領域を解放するガーベジコレクション方法であって、

複数スレッドそれぞれを順に選択する選択ステップと、

20 選択されたスレッドについて、当該スレッドの実行を停止して、当該スレッドからオブジェクトポインタの参照を通じてアクセス可能であるオブジェクトを検出して当該検出したオブジェクトを非解放対象として管理して、当該スレッドの実行を再開するという手順からなる検査処理を実施する検査ステップと、

25 前記選択ステップによる前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象として管理する検知ステップと、

前記複数スレッド全てについて前記検査処理が完了した後において、非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を、解放する解放ステップとを含む

30 ことを特徴とするガーベジコレクション方法。

1 2. 前記コンピュータは、メモリ管理ユニット（MMU）を用いてメモリを管理するメモリ管理機構を利用するものであって、オブジェクト指向プログラムの実行過程においてオブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保するものであり、

5 前記解放ステップは、前記メモリ管理機構を介して、メモリ領域の解放を行うことを特徴とする請求の範囲第11項記載のガーベジコレクション方法。

1 3. 複数スレッドで構成されるオブジェクト指向プログラムの実行過程において不要になったオブジェクトに対応するメモリ領域を解放するガーベジコレクション処理をコンピュータに実行させるためのコンピュータプログラムであって、

10 前記ガーベジコレクション処理は、

複数スレッドそれぞれを順に選択する選択ステップと、

選択されたスレッドについて、当該スレッドの実行を停止して、当該スレッドからオブジェクトポインタの参照を通じてアクセス可能であるオブジェクトを検出して当該検出したオブジェクトを非解放対象として管理して、当該スレッドの実行を再開するという手順からなる検査処理を実施する検査ステップと、

15 前記選択ステップによる前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象として管理する検知ステップと、

20 前記複数スレッド全てについて前記検査処理が完了した後において、非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を、解放する解放ステップとを含む

ことを特徴とするコンピュータプログラム。

1 4. 前記コンピュータは、メモリ管理ユニット（MMU）を用いてメモリを管理するメモリ管理機構を利用するものであって、オブジェクト指向プログラムの実行過程においてオブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保するものであり、

25 前記解放ステップは、前記メモリ管理機構を介して、メモリ領域の解放を行うことを特徴とする請求の範囲第13項記載のコンピュータプログラム。

図1

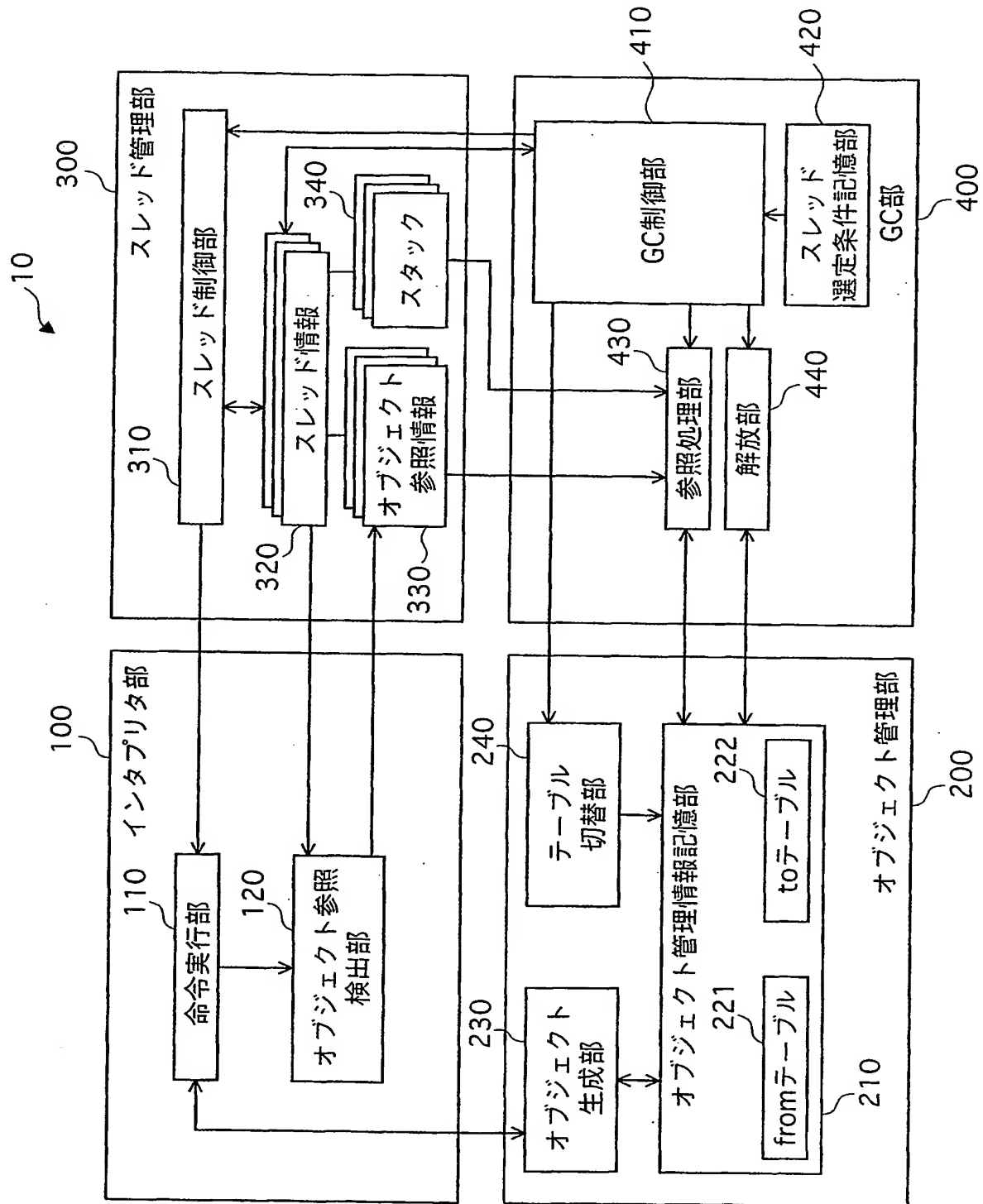


図2

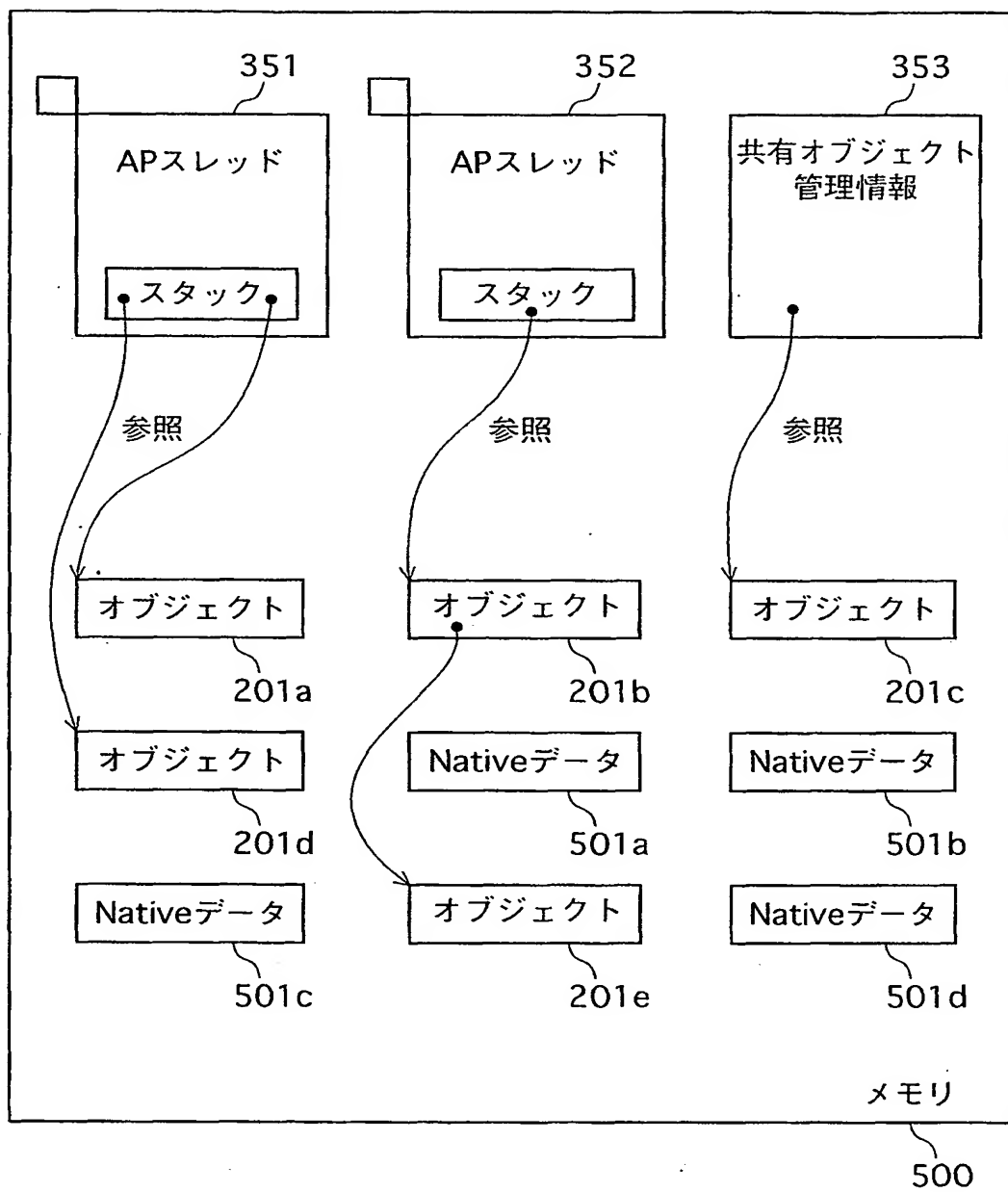


図3

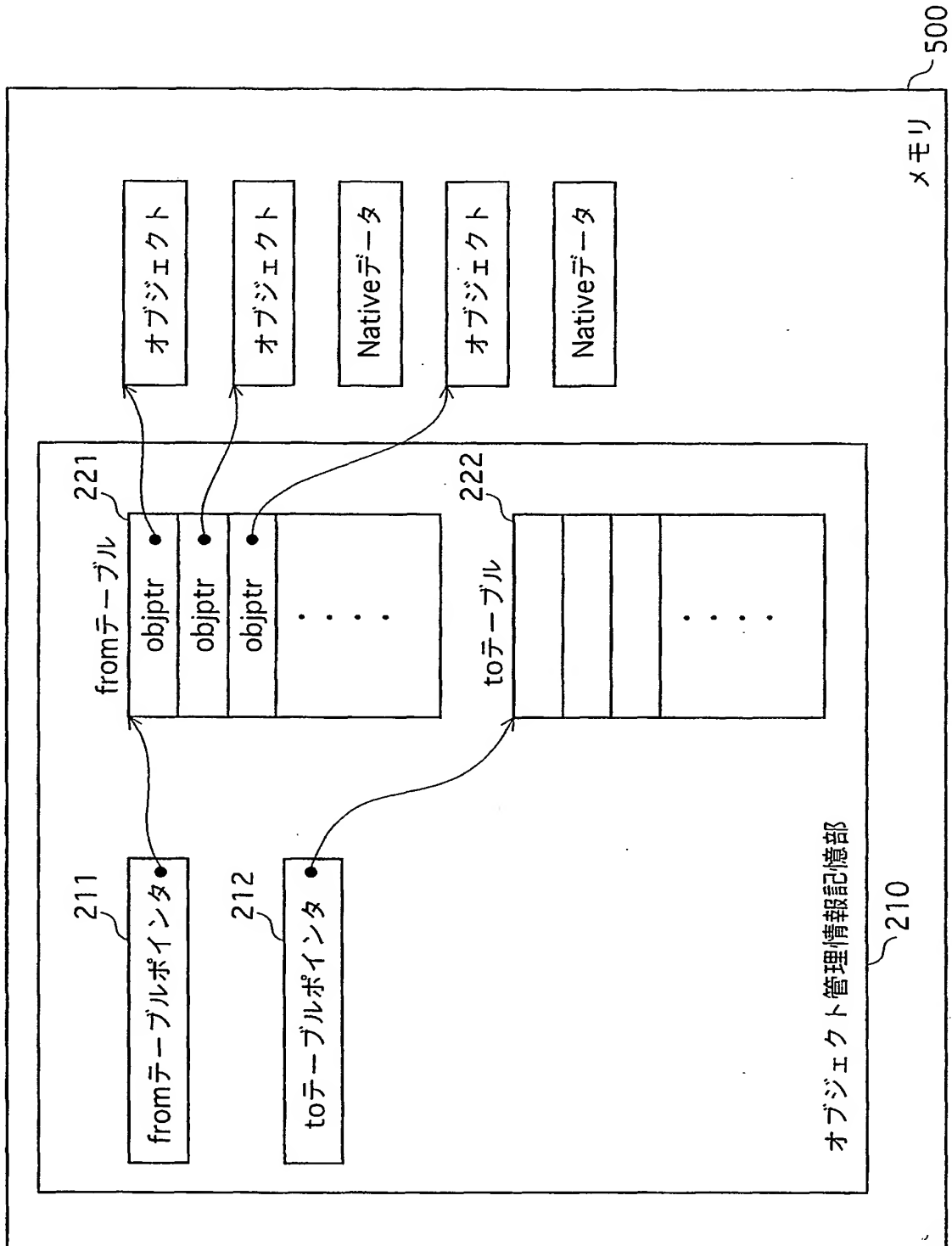


図4A

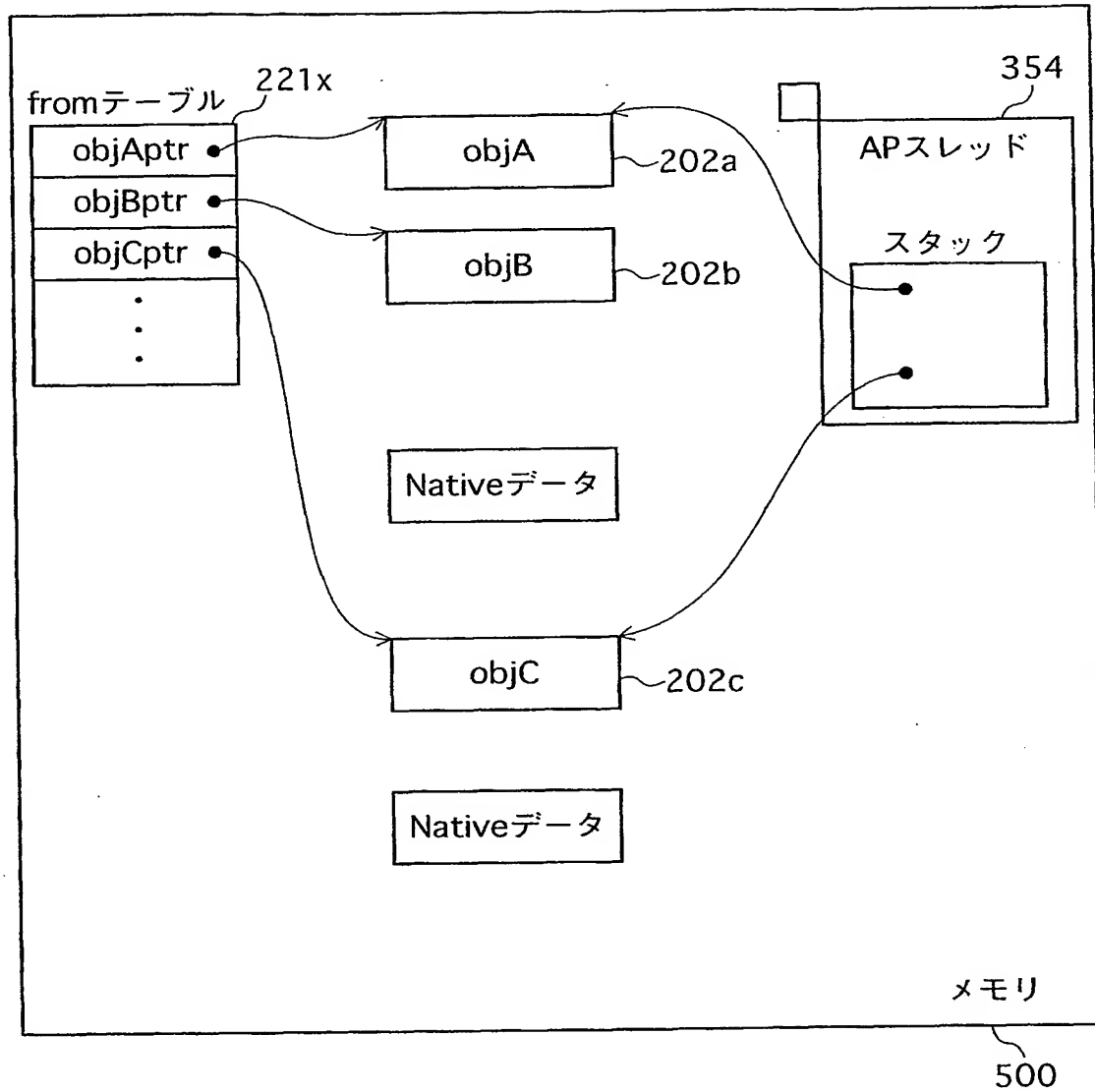


図4B

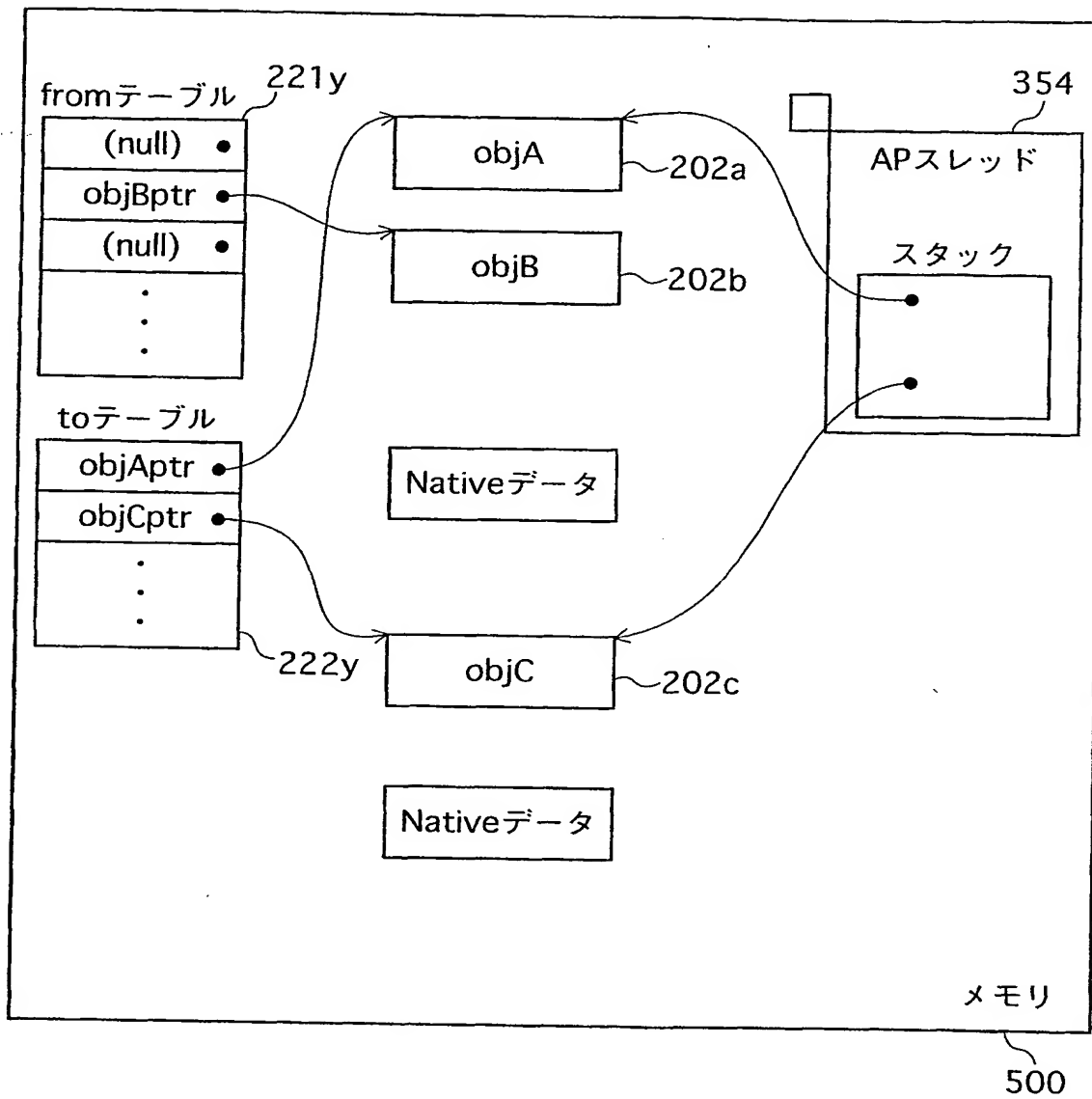


図5

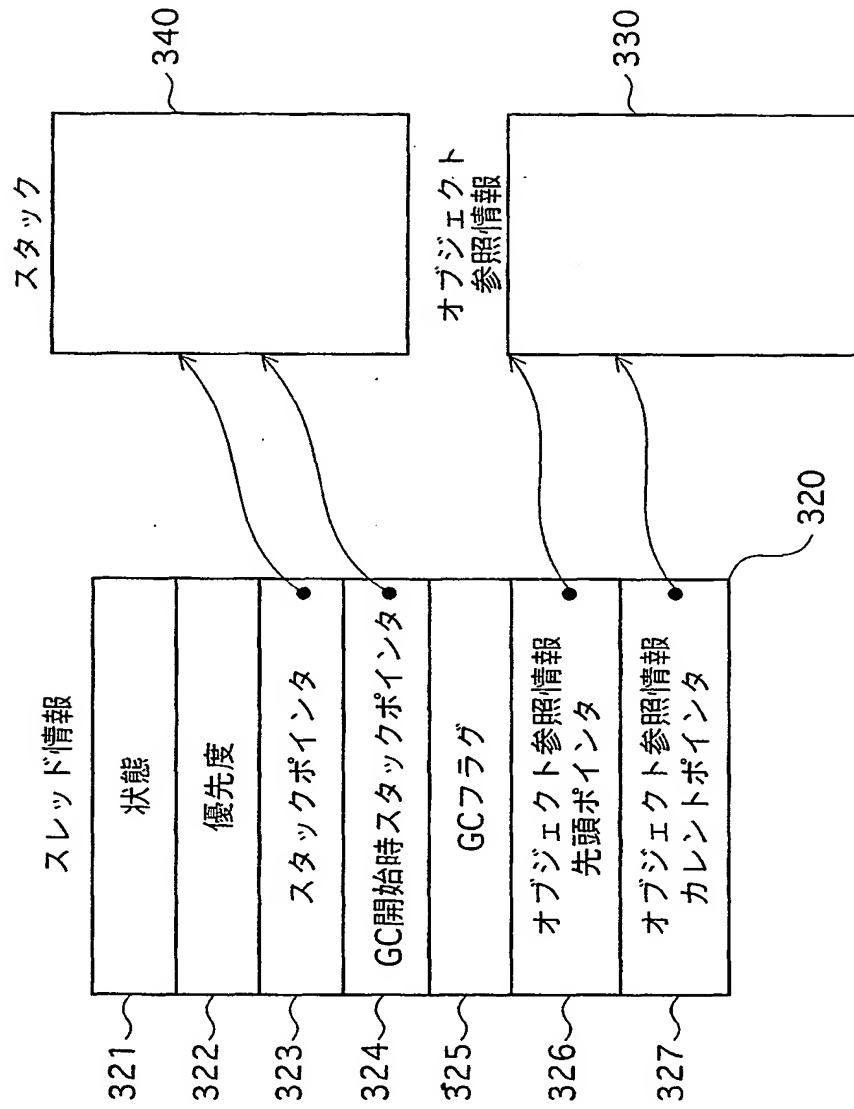


図6

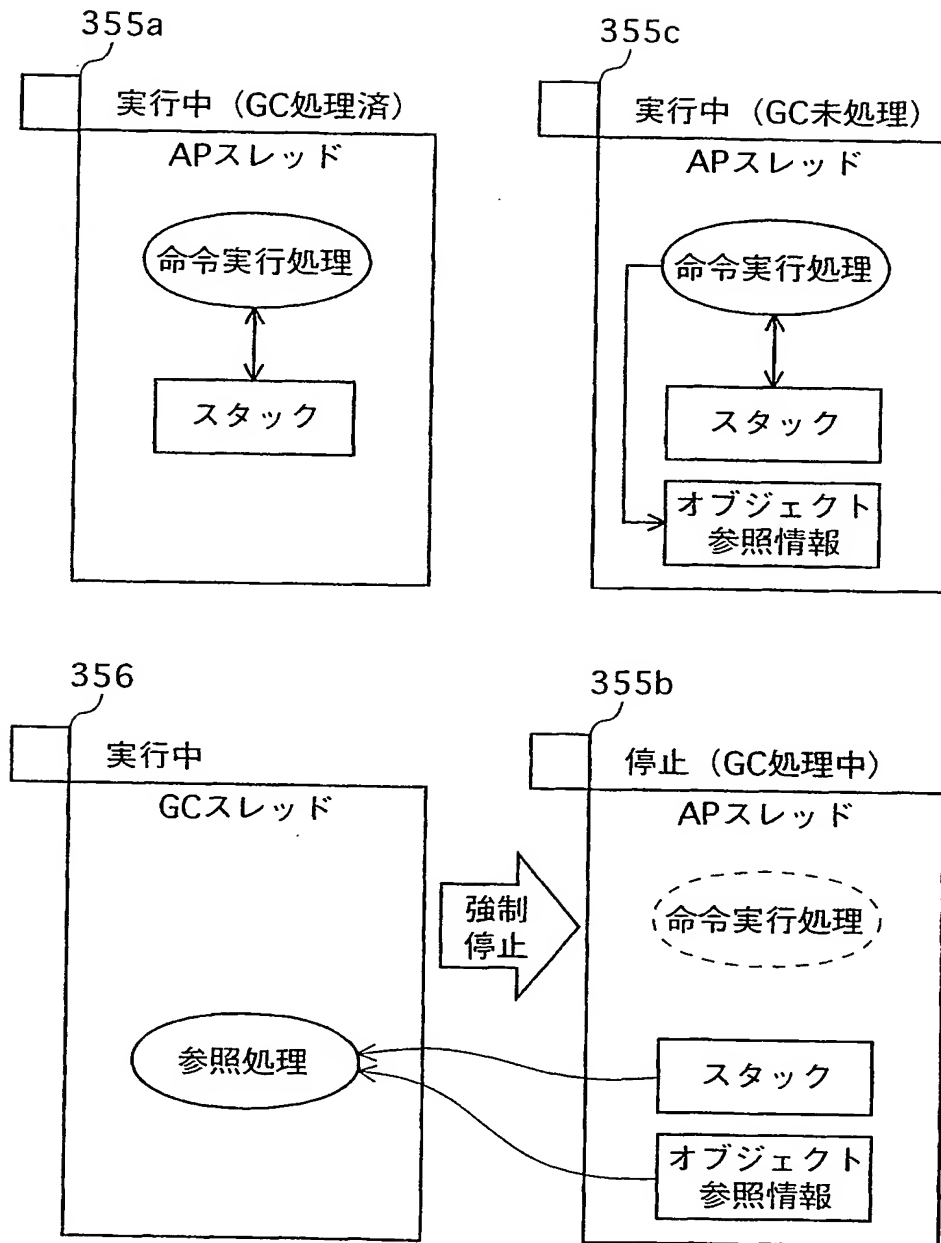


図7

スレッド選定条件

422	スレッド状態	WAIT 状態
423	スレッド優先度	低い
424	スタックサイズ	小さい

421

図8

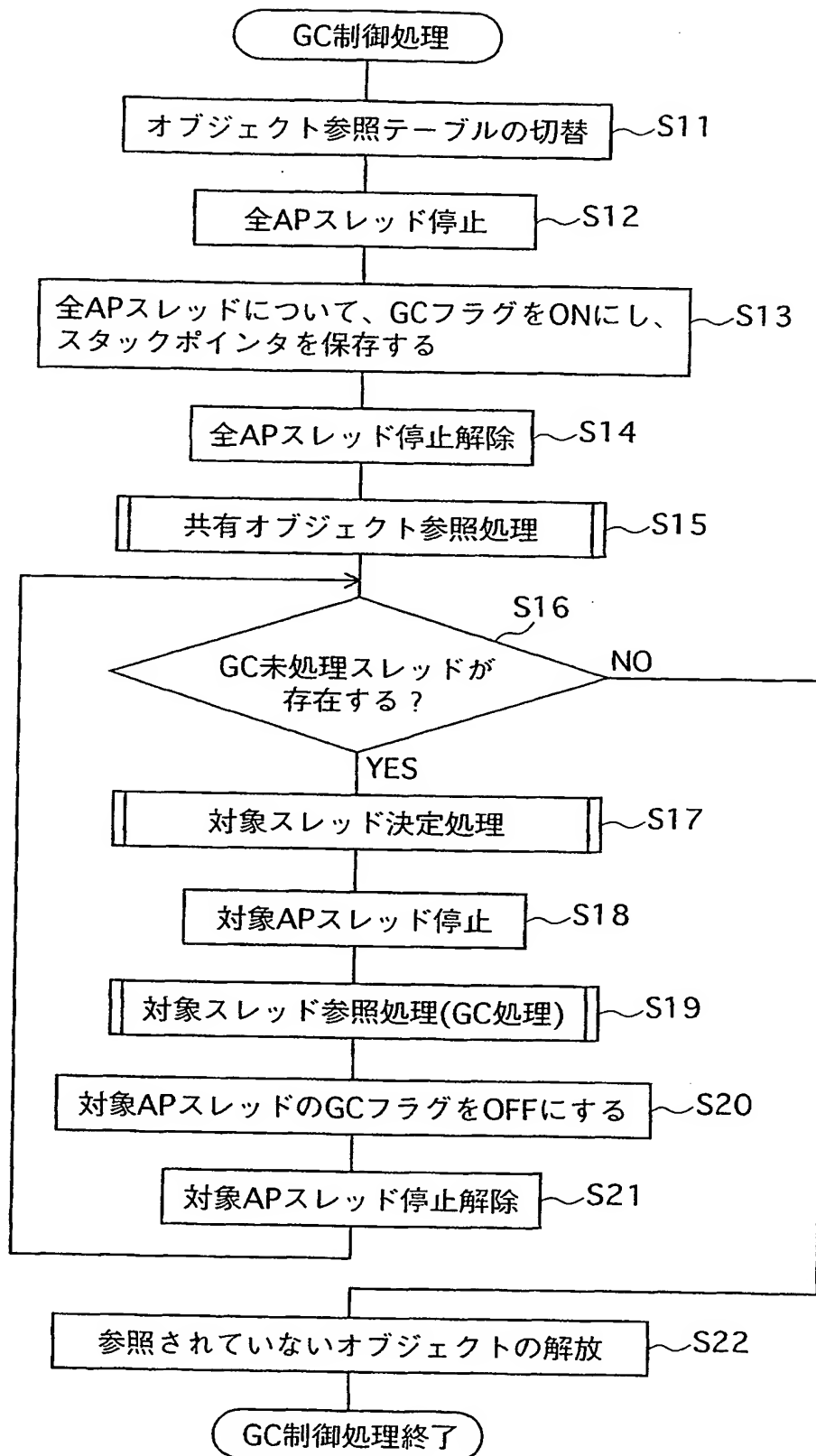


図9

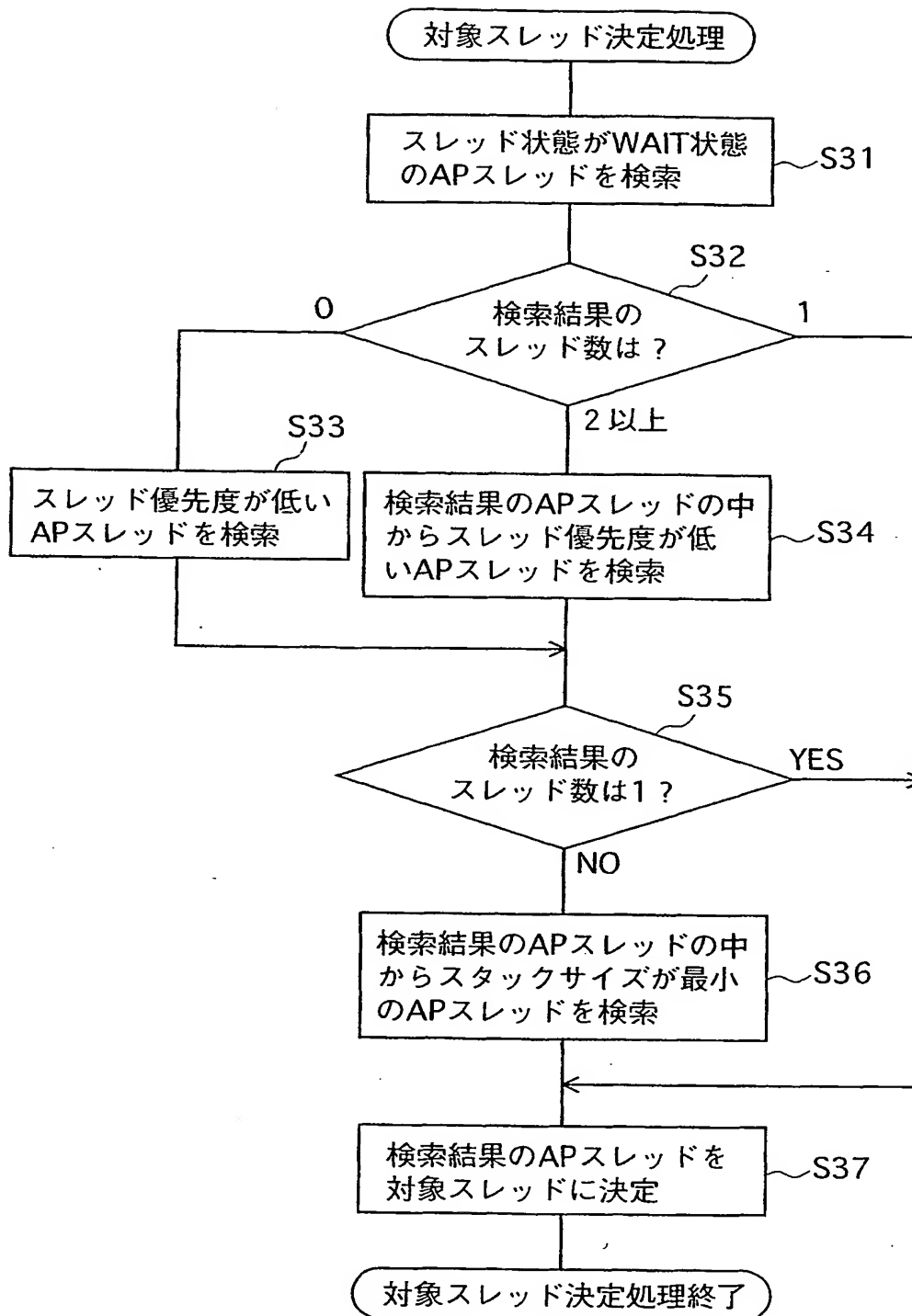


図10

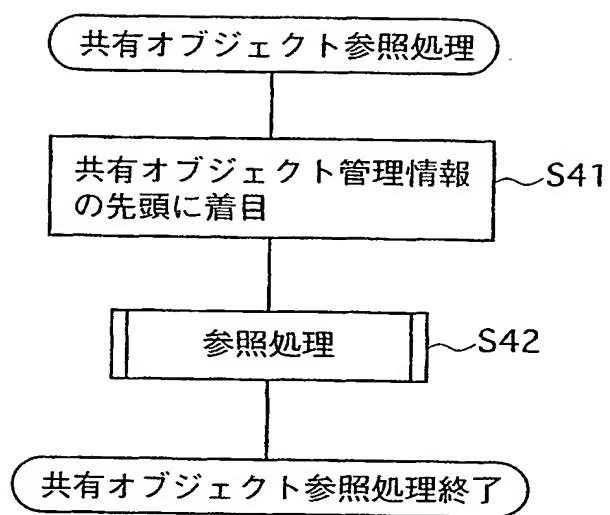


図11

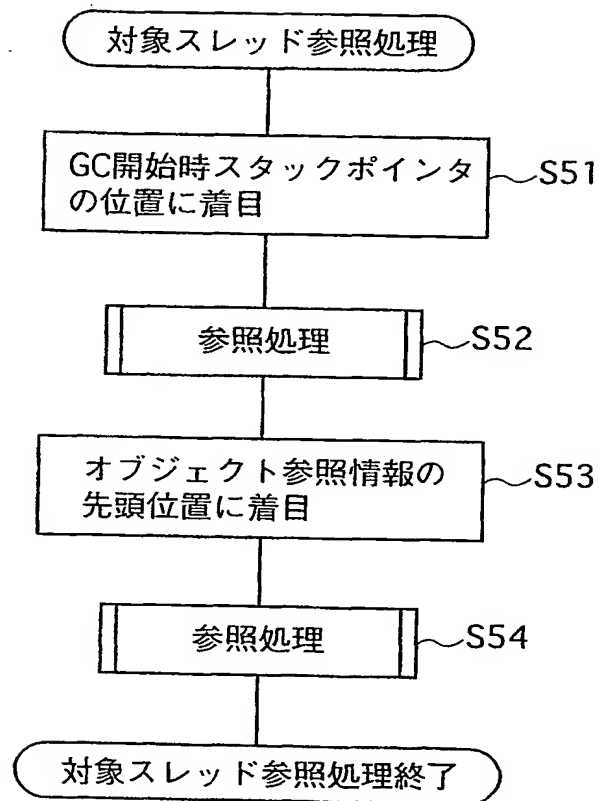


図12

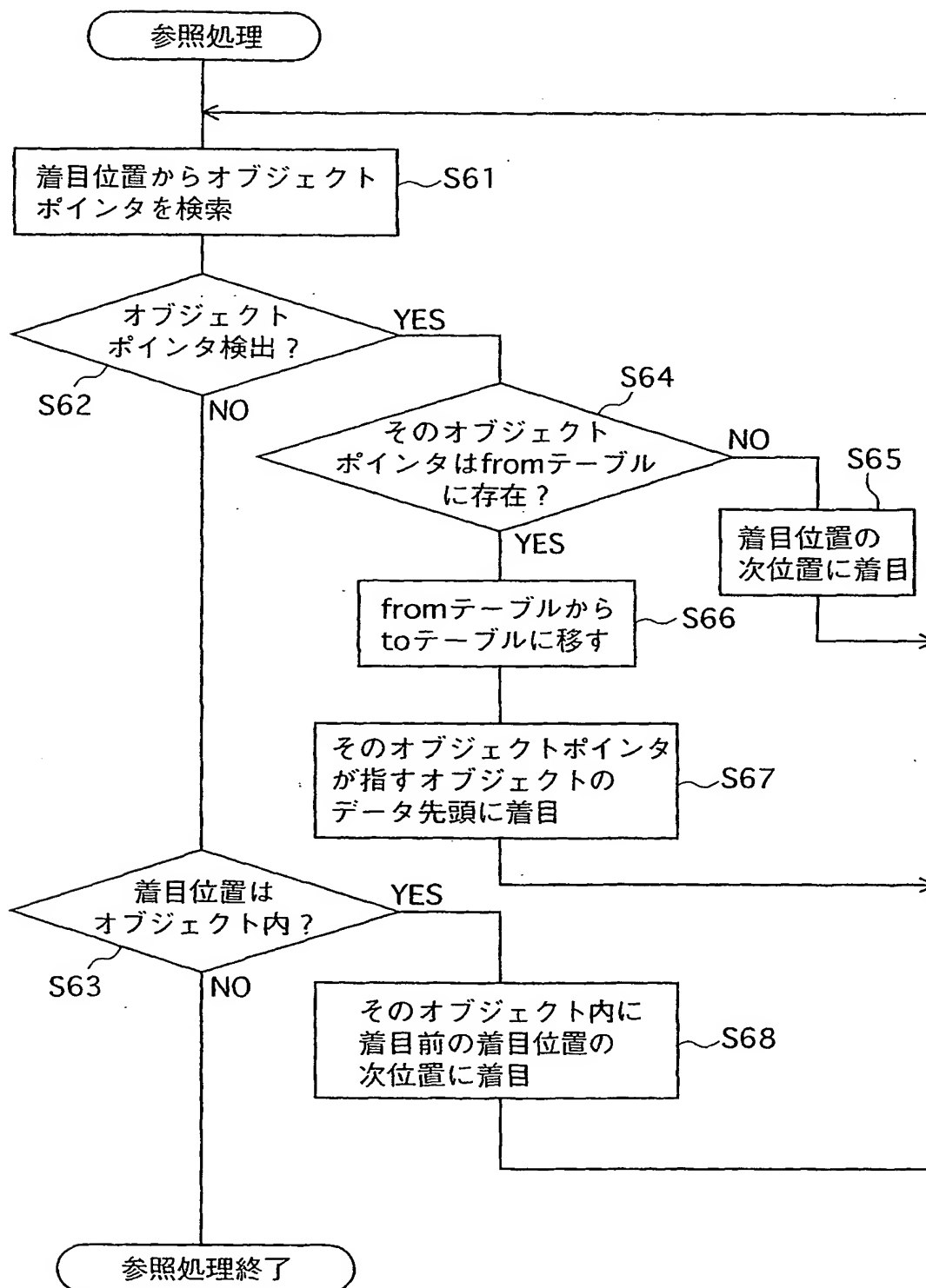


図13

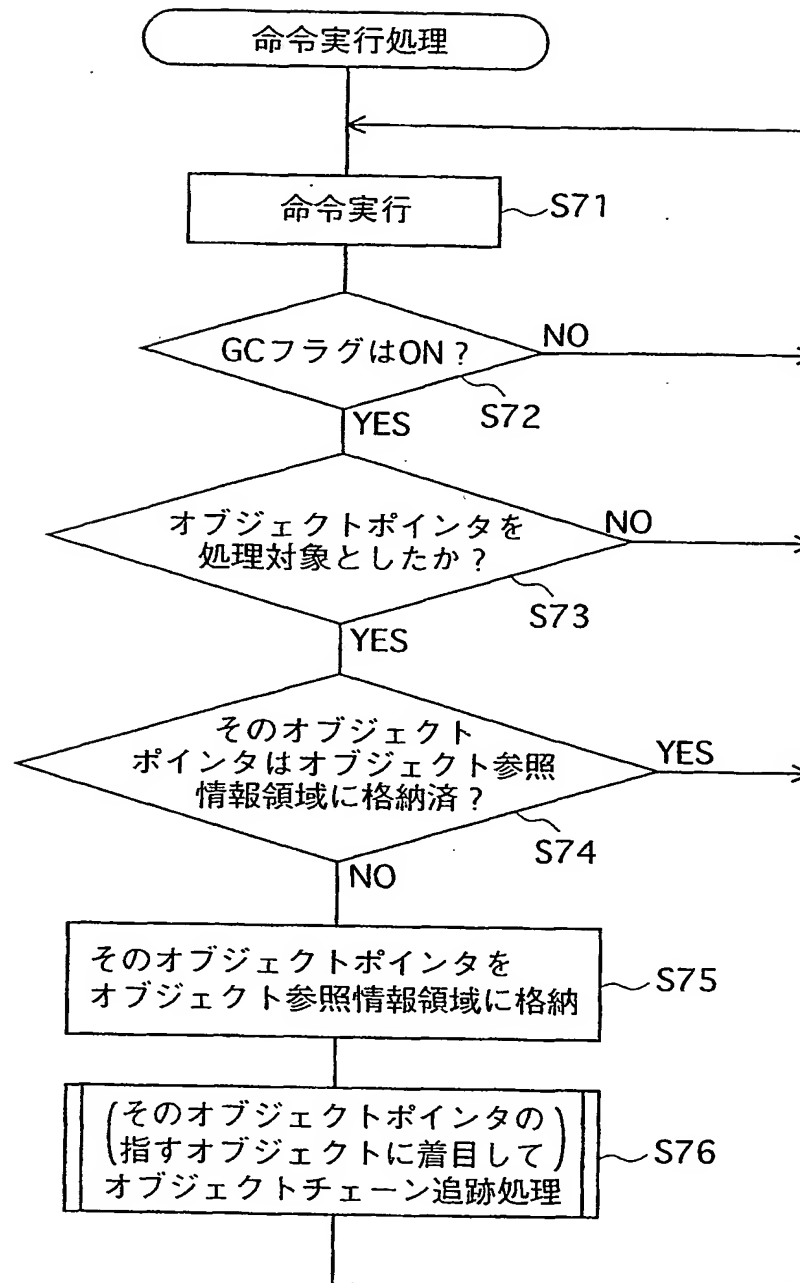


図14

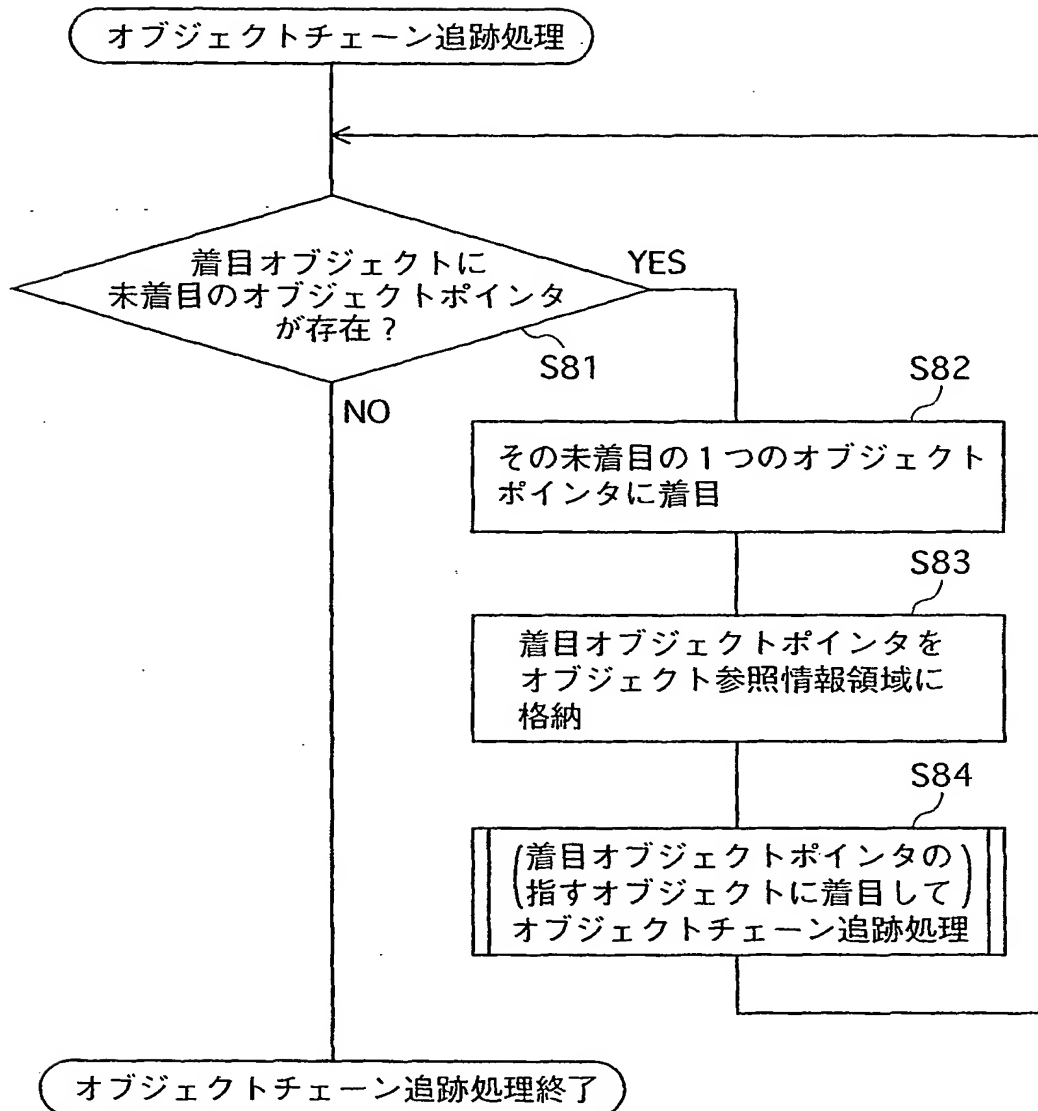


図15

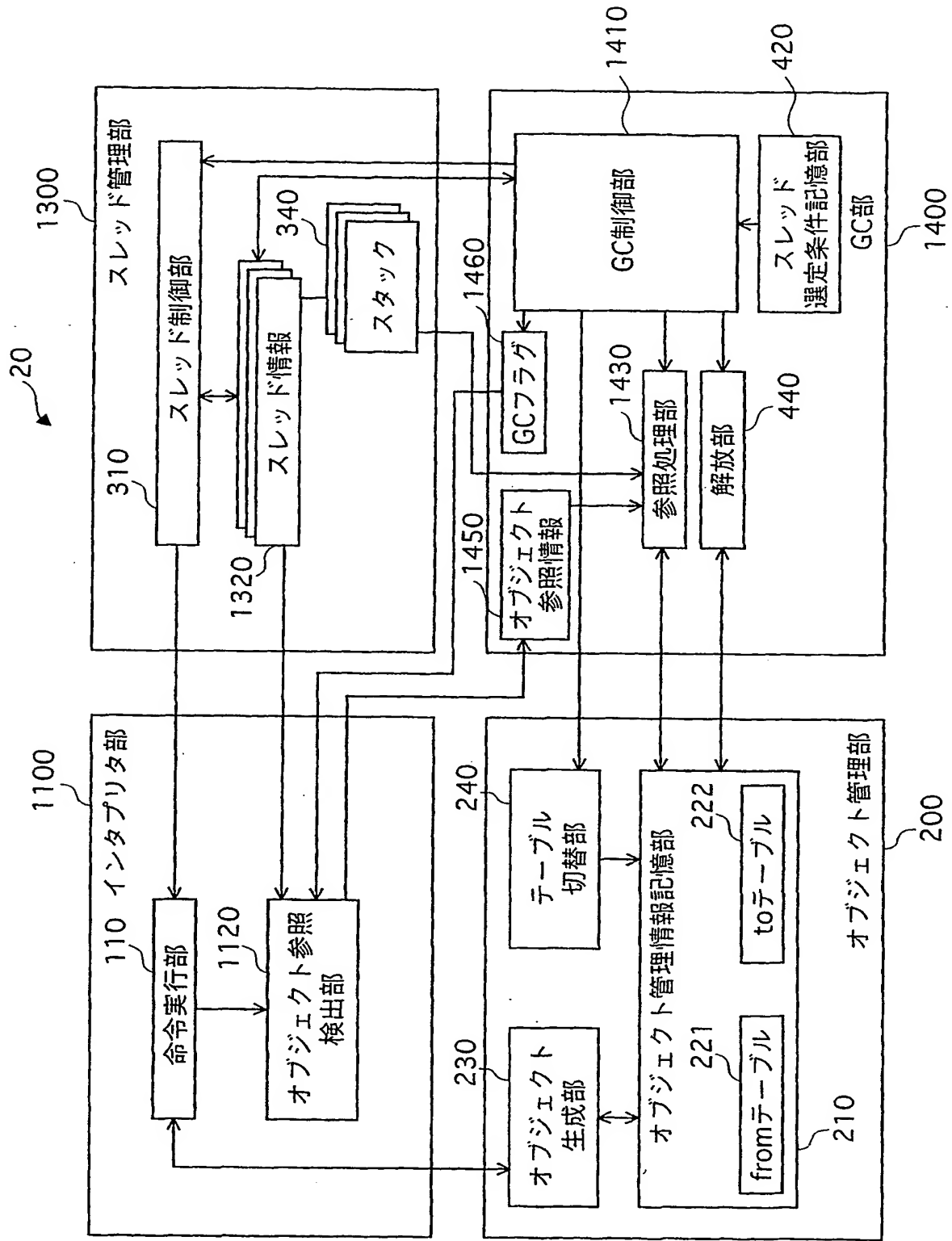


図16

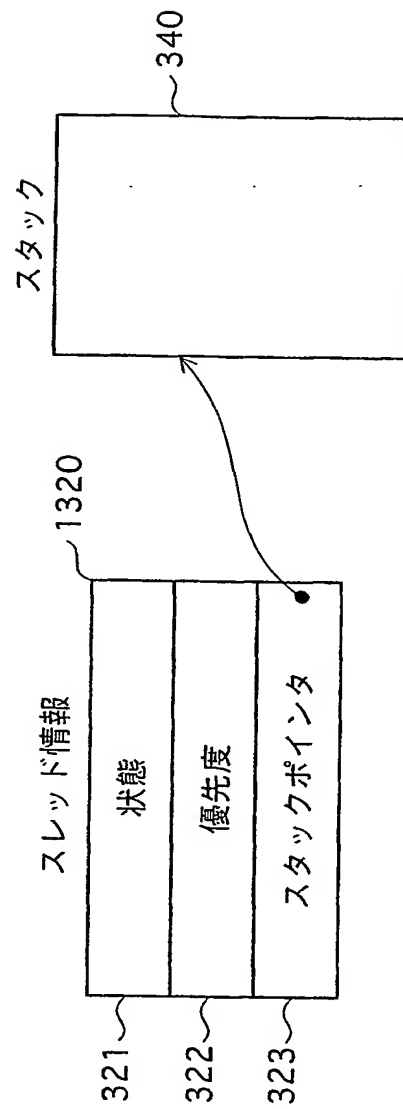


図17

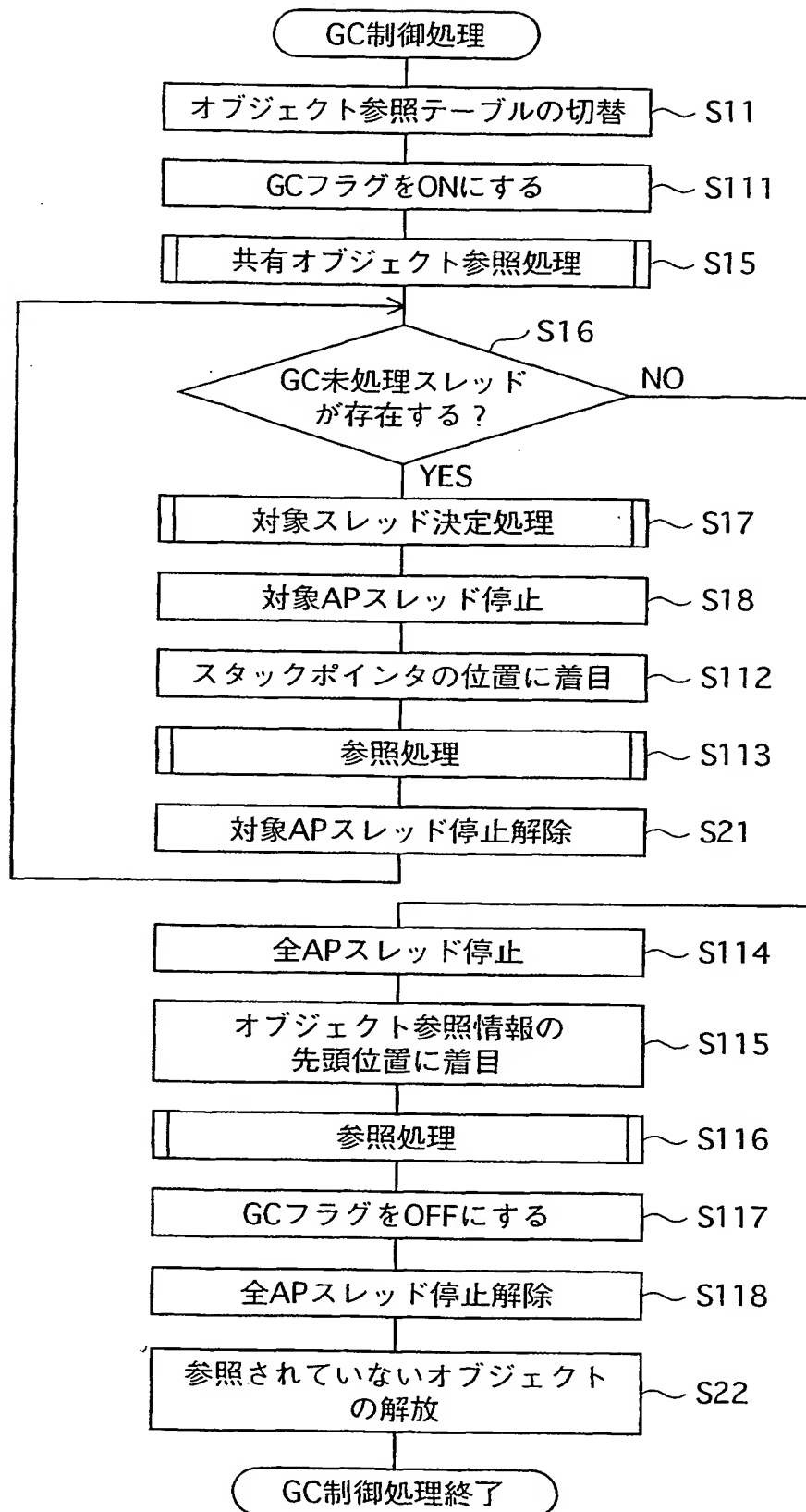


図18

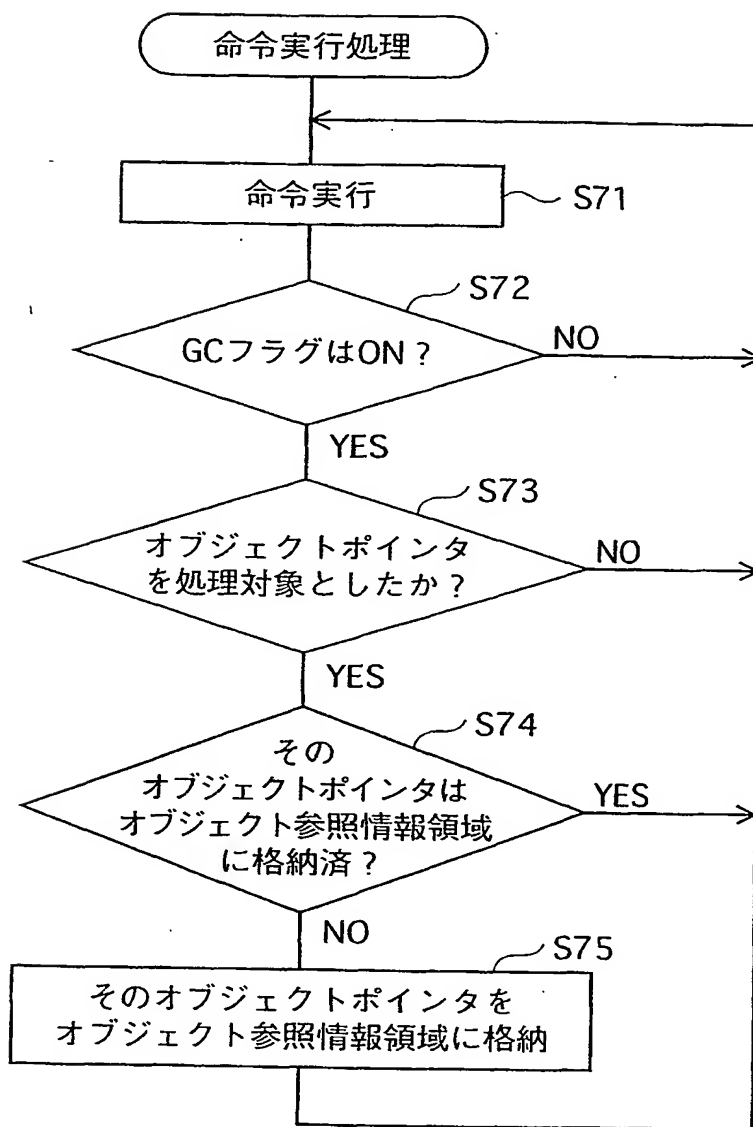


図19

